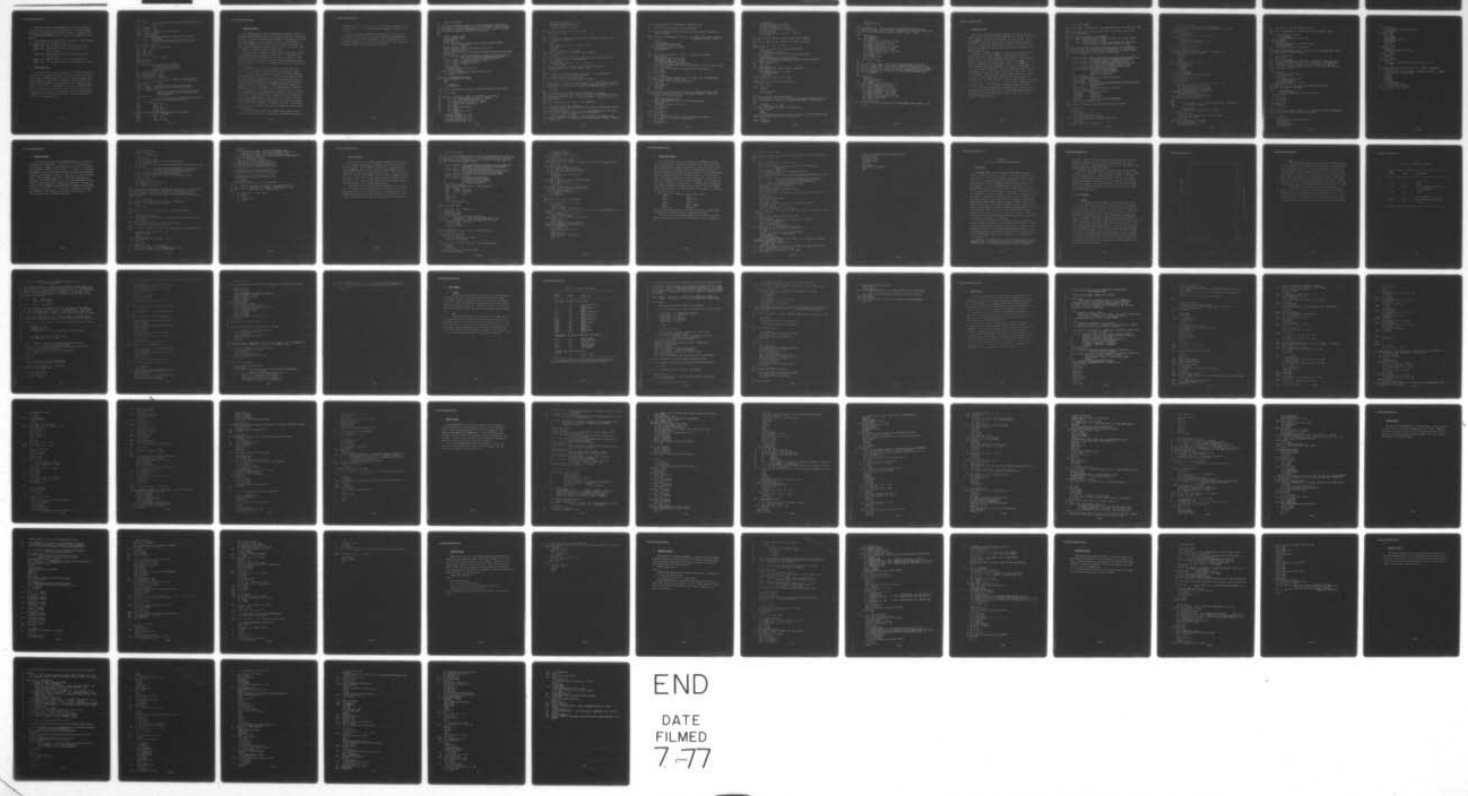
AD-A040 807 BDM CORP VIENNA VA F/G 15/5
MODELS OF THE US ARMY WORLDWIDE LOGISTIC SYSTEM (MAWLOGS). VOLU--ETC(U)
JAN 77 DAAG39-76-C-0134
UNCLASSIFIED BDM/W-76-211-TR-VOL-4A NL

| OF |
ADA
040807



END

DATE
FILMED
7-77

THE BDM CORPORATION

(1)

THE BDM CORPORATION

1920 Aline Avenue
Vienna, Virginia 22180
Phone (703) 821-5000

(12) 89p.

(11) Jan[redacted]77

(14) BDM/W-76-211-TR-VOL-4A

(15) DAAG39-76-C-0134

(6)

MODELS OF THE US ARMY WORLDWIDE LOGISTIC SYSTEM (MAWLOGS)

VOLUME IVA ADDENDUM TO PROGRAMMER'S GUIDE

4/B

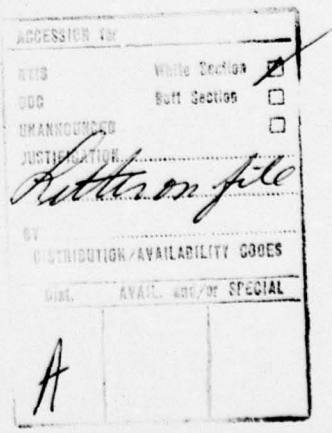
391 962

FOREWORD

This Addendum to the MAWLOGS Programmer's Guide is submitted to the Department of the Army, Washington, D.C. 20310 by the BDM Corporation, 1920 Aline Avenue, Vienna, Virginia 22180, as required by Contract Number DAAG39-76-C-0134.

This document is one of sixteen that describes the Models of the US Army Worldwide Logistic System (MAWLOGS). MAWLOGS was developed for the Deputy Chief of Staff for Logistics, Department of the Army, under the monitorship of the US Army Logistics Evaluation Agency and the US Army Logistics Center. The development objective was to provide a capability to analyze and compare the performance of multifunctional logistic systems, to include both current and proposed systems. MAWLOGS is not a model of a particular Army logistic system. It is a system for the rapid assembly of discrete-event stochastic simulation models of a wide range of logistic systems and for the processing and interpretation of data associated with the execution of such models. The original documentation was completed in 1974. Documentation for subsequent software development has added five volumes to the original eleven. The documents describing the system and how to apply it are listed below.

Volume I - General Description
Volume II - User's Manual
Volume IIA - Addendum to User's Manual
Volume III - Module Catalog
- Part 1 - Service Modules
- Part 2 - Field Maintenance and Supply Modules
- Part 3 - Wholesale Supply and Maintenance Modules
- Part 4 - Transportation Modules
- Part 5 - Communication Modules
- Part 6 - Continuous Service Modules
- Part 7 - Continuous Supply Modules
- Part 8 - Containerization Modules
- Part 9 - Model Change Modules



Volume IV - Programmer's Guide

Part 1 - Writing and Testing Modules, Module Library Maintenance and General Guidance

Part 2 - Technical Description of the Model Assembler Program

Part 3 - Technical Description of the Output Data Postprocessor System and Programs

Part 4 - Module Library Program Listings

Volume IVA - Addendum to Programmer's Guide

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I
INTRODUCTION

The MAWLOGS Programmer's Guide[1] is written at the computer programmer's level of detail, describing how to write and test MAWLOGS modules, the Model Assembler program, and the Output Data Postprocessor System (ODPS) programs. This addendum to Volume IV is provided to describe the changes required to add the continuous flow modeling capability to the MAWLOGS modeling system. Chapter II describes the new aspects of writing continuous flow modules, such as those described in Part 6 and Part 7 of the Module Catalog. Chapter III describes the changes made to the Model Assembler program and provides listings of the subroutines that have been changed. Chapter IV describes the ODPS programs in a similar fashion. The reader is reminded that this is an addendum to the Programmer's Guide and therefore should be used in conjunction with Volume IV and the various parts of Volume III, the Module Catalog.

---

[1] R. L. Kessinger, et al, Models of the U.S. Army Worldwide Logistic System (MAWLOGS), Volume IV, Programmer's Guide, General Research Corporation, OAD-CR-41, August 1974.
AD 923126, ...127, ...132

CHAPTER II
WRITING CONTINUOUS FLOW MODULES

A. BASIC DIFFERENCES

The basic difference in discrete event and continuous flow models requires significant differences in the programming of modules for the different families. A discrete event model represents only significant events in the system occurring at random times. In a continuous flow model, a rigid time stepping procedure is set up to update all variables in the system at the same time.

Although continuous flow verbs are designed to be utilized in a model description in the same manner as those in the discrete event families, the programming structure of continuous flow verbs is quite different. A typical discrete event verb represents the processing of a demand or shipment which is characterized by a set of attributes in the arrays ATRIB or HOLD. A verb expects certain attributes to be used for specified purposes and will alter those attributes and pass their contents back to the calling program or store them in a file for later processing. The set of attributes is sent through the node network system by references to different nodes and exists as an entity throughout its lifetime, either in a verb during processing, in the time file waiting for the next event to occur, or in a queue at a node.

In contrast, continuous flow models represent the rates of flow between nodes and the levels of materiel at nodes in the system. The continuous flow verbs contain equations to calculate these rates and levels at every time step, based on values from the previous time step. There is no identifiable set of attributes representing an individual demand or shipment traveling through the system. The level of detail in a continuous flow module is determined by what types of rates and levels are represented and what interactions between them simulated. Since a module does the same thing every time step, the size of the time step in a model determines how

closely the model represents the real system. For a steady state system with a slow rate of change, a large time step is possible, thereby reducing computer running time. For a system in a state of flux with rapidly changing rates, a small time step is required in the model to properly represent the system flows.

B. MODULE STRUCTURE

A continuous flow model is designed so that every module is executed twice in every time step, first to update the levels of materiel, and a second time to calculate all the new rates of flow in the system. The model also operates in two modes, one to set up the rate/level linkage between nodes and the other to simulate the flows in the system. Due to this method of operation there is a general structure used in every continuous flow module. Figure II-1 outlines this structure. There is a setup section which is executed in one of the passes when the model is operating in setup mode. Then there is a section of code which are the equations to update the levels represented in the module. This section is executed during the levels pass in the simulation mode. Once the section of code is completed, control is returned to the calling program by a RETLOG. Finally, there is a section of code which contains the equations to update rates of flow which is executed during the rates pass in the simulation mode. Each section is independent of the other and no two of them are ever executed together during the same pass. Each module has this structure, but some may have vacuous sections. For example, a module may contain equations to update rates only, and return control immediately when it is called during a levels pass.

Within this structure, parameter slots are utilized in the standard MAWLOGS fashion to provide the flexibility when a module is coded of not specifying what programs must be used with it. Thus, a certain procedure to be simulated need not be specified until the model description is written with the name of the procedure module entered in the parameter slot.

```
      /* CFVRB, S, O-
* CALL RLSYS
      IF (RLMODE .EQ. SIM) GO TO 100
```

```
C  SETUP - DEFINE LINKS
      IF (RLSW. EQ. LEVELS) CALL RETLOG
            .
            .
            .
      CALL RETLOG
```

```
C  SIMULATION
100  IF (RLSW. EQ. RATES) GO TO 150
C  LEVELS PASS
      EQUATIONS TO UPDATE LEVELS
            .
            .
            .
      CALL RETLOG
```

```
C  RATES PASS
      EQUATIONS TO UPDATE RATES
            .
            .
            .
      CALL RETLOG
```

Figure II-1. Structure of Continuous Flow Module Programs

C. STATISTICS COLLECTION

A programmer writing function modules of the rate/level variety need not be concerned with the collection of statistics in the program. In discrete event modules, the collection of statistics must be performed every time a variable is changed in the module program. However, this function is performed at prespecified intervals for all flagged variables by the module RLSTAT in continuous flow modules. This function and the associated service modules are described in detail in Part 6 of Volume III, the Module Catalog.

D. DATA STRUCTURE

The continuous flow modules that have been programmed to date are documented in the Module Catalog, Part 6, Continuous Service Modules, and Part 7, Continuous Supply Modules. These modules utilize the PDS dataset structure for storing and referencing permanent attribute data. In addition, there are two common blocks which are used for temporary attributes and working storage areas. The common block /RLSYS/ is utilized by the rate/level system routines. This block contains the rate/level control words such as the size of a time step, the identification of the mode of operation for a pass of the model, rate/level delay parameters, and rate/level PDS dataset type identifiers. The common block /SUPC/ is utilized by the continuous supply modules. It contains variables that are set by modules which loop through various types of PDS datasets in the model. These variables include stock status, rates of flow, fill rates, and splitting factors which are in turn referenced by individual supply modules in calculating new rates and levels. Each of these common blocks is described in detail in their respective parts of the Module Catalog.

CHAPTER III
MODEL ASSEMBLER PROGRAM CHANGES

A. INTRODUCTION

The MAWLOGS Model Assembler Program was designed to read a coded description of a node network structure, select the necessary pre-programmed modules from the Module Library, and create a complete simulation model program of the logistics system. The use of the Model Assembler is described in the User's Manual[2,3] and a complete technical description is given in the Programmer's Guide[4].

The changes made in the Model Assembler have been minimal. These changes can be placed in three categories: (1) expansion of the verb argument types accepted, (2) reduction of Model Assembler printed output, and (3) increase in the number of modules allowed in the library and in a model description. Program changes have been made in six subroutines of the Model Assembler (BLOCK DATA, HTSCAN, MDSCAN, NODEND, NODFMT, PSTRIN). The description and current listings of these routines are included in this chapter. These descriptions supercede the respective listings in the original Programmer's Guide.

The types of verb arguments now accepted by the Model Assembler are positive integers, negative integers, positive floating point, negative floating point, and hollerith. The exact format is given in the Addendum to the User's Manual.

---

[2]Burger, R. T., et al, Models of the U.S. Army Worldwide Logistic System (MAWLOGS), Volume II, User's Manual, General Research Corporation, OAD-CR-41, August 1974.

[3]Burger, R. T., Models of the U.S. Army Worldwide Logistic System (MAWLOGS), Volume IIA, Addendum to User's Manual, The BDM Corporation, BDM/W-76-211-TR, January 1977.

[4]Burger, R. T., Models of the U.S. Army Worldwide Logistics System (MAWLOGS), Volume IV, Programmer's Guide, Part 2, Technical Description of the Model Assembler Program, General Research Corporation, OAD-CR-41, June 1974.

The printed content of the HOLDTABLE for a node is now suppressed unless there is an error in the node description. This greatly reduces the number of pages produced by the Model Assembler. An example of the new format of the HOLDTABLE listing is shown in the Addendum to the User's Manual.

Certain dimensions have been expanded in the Model Assembler Program. The current capacity of the program is as follows:

NVMAX = 600, Maximum number of modules on the module library

NSVMAX = 400, Maximum number of simple verbs referenced in a model description

NVNMAX = 300, Maximum number of modules referenced in a single node description

NODMAX = 100, Maximum number of nodes in a model description

NAMMAX = 60, Maximum number of common decks referenced in a model.

B. PROGRAM BLOCK DATA

This program contains the data statements to initialize the necessary common variables in the Model Assembler program. The name of the module which keys the standard package of routines for a model are defined here. The name AAMAIN is placed in the first portion of the array SIMVB. The pointer to the array SIMVB, the variable ISIM, is set to the location of AAMAIN. The array ISI is initialized with the value 2 in the first ISIM positions to indicate that the module is a routine. The characteristic for the module (P) is entered in array NPSLOT. The cross reference cards in AAMAIN and in those modules referenced by it directly and indirectly then define the standard package.

```
      BLOCK DATA
      COMMON /EXPA  / IDOL,ICOMMA,LPAR,IEQUAL,IRPAR,IPER,IAST,ISLASH,
     1                IPLUS,IMINUS
      COMMON /KN    /  KNODE
      COMMON /APHAS /  IFL
      COMMON /IPSVB /  IPSVB
      COMMON /MAIN/ AVB,IFREEH,IENDS,ANAND,ANULL
      COMMON / DTIM / KC,DTIM
      COMMON / IERCT/ IERCT
      COMMON /SCAN/ S(20,50),IPP(20,50),N(20,50),IP(20),JS(20)
     1              ,IMAX,JMAX
      COMMON /SIMVB / SIMVB(400),VREP(400),ISIM,KSIMVB(400),INDVB,
     1                INDLIB,NSVMAX,VRHOD(300),NVNOD,NVNMAX
      COMMON  /MHOLD/ MAXHLD
      DATA  MAXHLD  /2000/
      COMMON /WRT/  KPARIT,WR(40),KPMAX
      DATA  KPMAX   /40/
      DATA  IMAX  /20/
      DATA  JMAX  /50/
      DATA NVNMAX/300/
      DATA  IENDS  / 0/
       DATA             NVMAX    /600/
      DATA NSVMAX/400/
      DATA  IPSVB /0/
      DATA IFL   /1/
      COMMON /COMDEK/ COMNAM(60),INDNAM,NAMMAX,
     1                COMDIM(80,2),INDDIM,MAXDIM,
     2                NODNAM(100),INDNOD,NODMAX,
     3                INITCD(60),LASTCD(60)
      DATA  INDNOD,NODMAX  /0,100/
      DATA  INDNAM,NAMMAX  /0,60/
      DATA  INDDIM,MAXDIM  /0,80/
      DATA  INITCD,LASTCD  /60*0,60*0/
      COMMON /GCHARC/ NDEL,NEND,STRING(1),BUFF(80),LUNIT,INC,ICD,LOUT
     1               ,OLDBUF(80),LOCATE(10),CDSCAN,NEWCRD,NCHAR
      DATA  NEWCRD,CDSCAN  /0,0 /
      DATA  NCHAR          /10  /
      COMMON /SWITCH/ IXFL,LIBFL,KEYFMT,LSTMOD,ISTDSR,MUPDAT
      COMMON  /VLIB/  IV,VI(600),ISI(600),NPSLOT(600),NVMAX
      COMMON /NUMBRS/ DTA(21)
      DATA       DTA       /1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,
     1                      2H10,2H11,2H12,2H13,2H14,2H15,2H16,2H17,
     2                      2H18,2H19,2H20                          /
       DATA IERCT /0/
      DATA             IDOL,ICOMMA,LPAR,IEQUAL,IRPAR,IPER,IAST,ISLASH,
     1                 IPLUS,IMINUS/1,2,3,4,5,6,7,8,9,10/
      DATA    INC /128/
      DATA             KTIM,DTIM/0,0/
      DATA             KNODE / 0     /
      DATA             IV,IL /0,0    /
      DATA             ANULL /1H*    /
      DATA             IENDS   /0/
C.. INITIALIZE NAMES OF STANDARD SERVICE ROUTINE PACKAGE
      DATA             SIMVB   /6HAAMAIN/
      DATA             ISIM   /1/
      DATA             ISI   /2/
      DATA             NPSLOT   /1HP/
      END
```

C. SUBROUTINE HTSCAN

This routine performs a scan of the HOLD table developed in Phase I of node processing and builds stacks which correspond to the levels of parenthesization. The stacks contain the content words of the parameter slot strings for verbs referenced in the node and are utilized by routine WRISTA to create the FORTRAN code for the P10000 linkage routines for the node. Up to 20 levels are allowed including the main level of the node itself, that is, parameter slots may be nested to a depth of 19.

Stacks are built in a manner so that as each left parenthesis - except for a left parenthesis associated only with a "P=" - is encountered, the current stack is placed in abeyance and a new stack is started. As each right parenthesis is encountered, an opposite process occurs and the stack started last is written out by routine WRISTA and processing of the previous stack is resumed.

The stacks are built in array S(I,J) where I is the level of the stack and J is the index to the content words contained in the stack at level I. There are several corresponding arrays which contain additional data relating to the stack entries. Array N(I,J) contains packed information created during the HOLD table scan. The right hand portion of the word (packing factor 4096) contains the number of the subnode mentioned in a node reference or the number of the P10000 routine which handles the parameter slots for a verb reference. If no subnode number is mentioned in the model description, the default value set in N(I,J) is one. If there are no parameter slots specified with a verb reference, the value in N(I,J) will be zero. The left hand portion of the word in N(I,J) can contain the number of the subnode which the content word in S(I,J) begins, the number of the parameter slot which the content word in S(I,J) begins, or nothing if the content word is imbedded in a subnode or PS string. If the content word in S(I,J) is a node name rather than a verb name, the value in N(I,J) is multiplied by -1.

Array IP(I) holds the number of the P10000 routine which is written out from the stack for level I. The position in the HOLD table where the

argument string (i.e., P = i,j,k,...) for a verb reference begins is stored in array IPP(I,J). If a verb reference has no arguments, IPP(I,J) is set to zero.

As the stacks and corresponding arrays are built, the order of the delimiters and content words is carefully checked and error messages are printed out for any illegal combinations. One new combination has been added as acceptable, a minus sign in an argument string. The code has been expanded to handle this case at sections 20 and 40.

```
      SUBROUTINE HTSCAN
C..
C.. THIS ROUTINE CONTROLS PHASE 2 IN THE PROCESSING OF EACH NODE IN
C.. THE MODEL.  THE HOLD TABLE IS SCANNED AND A STACK IS SET UP FOR
C.. EACH LEVEL IN THE NODE DESCRIPTION. THE ROUTINE WRISTA IS CALLED
C.. TO WRITE THE LINKAGE ROUTINES FOR EACH STACK
C..
C..
       COMMON /IERSW/ IERSW
      COMMON /IPSVB/ IPSVB
      DATA IPSVB /0/
      COMMON /I/ I,J
      COMMON /SCAN/ S(20,50),IPP(20,50),N(20,50),IP(20),JS(20)
     1              ,IMAX,JMAX
      COMMON /ADDIT/   IRET
      COMMON /ADVSCA/ ISCAN,INEND,IPDELW,PCONTW,IVR
      COMMON /ER/ ICOH,ANM1
      COMMON /EXPA/ IDOL,ICOMMA,LPAR,IEQUAL,IRPAR,IPER,IAST,ISLASH,IPLU
     1S,IMINUS
      COMMON /GCHARC/ NDEL,NEND,STRING(1),BUFF(80),LUNIT,INC,ICD,LOUT
     1               ,OLDBUF(80),LOCATE(10),CDSCAN,NEWCRD,NCHAR
      COMMON  /VLIB/  IV,VI(600),ISI(600),NPSLOT(600),NVMAX
      COMMON /MAIN/ AVB,IFREEH,IENDS,ANWND,ANULL
      COMMON /TABLE / IBEG(20),ILEN(20)
      COMMON  /MHOLD/  MAXHLD
      COMMON HOLD(1)
      EQUIVALENCE (HOLD(1),IHOLD(1))
      DIMENSION IHOLD(1)
C.. ERROR, DON?T PROCESS HOLD TABLE IF IENDS .EQ. 9
      IF(IENDS.EQ.9) RETURN
C..
C.. BEGIN PROCESSING HOLD TABLE
C..   INITIALIZE STACK POINTERS
      I=1
      J=0
          ISCAN=0
      IHOLD(1)=ICOMMA
C..
C.. RETRIEVE NEXT DELIMITER CODE AND CONTENT WORD FROM HOLD TABLE
C..
    1 CALL ADVSCA
C..
C.. STATEMENT NUMBERS OF  LOGIC FOR DIFFERENT DELIMITERS
C..       10 - LEFT PARENTHESIS FOLLOWED BY INTEGER
C..       20 - LEFT PARENTHESIS FOLLOWED BY   P
C..       30 - DOLLAR SIGN FOLLOWED BY INTEGER
C..       40 - DOLLAR SIGN FOLLOWED BY   P
C..       50 - COMMA
C..       60 - RIGHT PARENTHESIS
C..       70 - SLASH
C..       80 - DOLLAR SIGN ALONE
C..       90 - PERIOD
      IF(IPDELW.GT.900) GO TO 10
      IF(IPDELW.EQ.899) GO TO 20
      IF(IPDELW.GT.100) GO TO 30
      IF(IPDELW.EQ.99) GO TO 40
      IF(IPDELW.EQ.ICOMMA) GO TO 50
```

```
      IF(IPDELW.EQ.IRPAR) GO TO 60
      IF(IPDELW.EQ.ISLASH) GO TO 70
      IF(IPDELW.EQ.IDOL) GO TO 80
      IF(IPDELW.EQ.IPER) GO TO 90
C
C  FALL THRU IF IPDELW NONE OF THE ABOVE
      CALL ER(100,0,1)
      GO TO 1
C..
C.. DELIMITER IS A COMMA, ANOTHER ENTRY TO BE MADE IN CURRENT STACK
C..      INCREMENT POINTER J
50    CONTINUE
      J=J+1
      IF (J .GT. JMAX) CALL ER (7,5,0,PCONTW)
      IPP(I,J)=0
      N(I,J)=0
      IF(PCONTW.EQ.ANULL) GO TO 501
C.. ENTER CONTENT WORD FOUND AFTER COMMA IN STACK
502   S(I,J)=PCONTW
      GO TO 1
C.. NULL CONTENT WORD AFTER COMMA, CHECK FOR NEXT DELIMITER, SHOULD
C..   BE AN ASTERISK FOR A NODE REFERENCE
501   CALL ADVSCA
      IF(IPDELW.NE.IAST) CALL ER(      101,0,0)
         N(I,J) =-1
C  MINUS N() VALUE FLAGS  SUBNODE ENTRY NO, FOR NODE REF,
C   POSITIVE FLAGS POS NO FOR PS STRING REF
      GO TO 502
C..
C.. DELIMITER IS A PERIOD, SHOULD ONLY FOLLOW A NODE REFERENCE
C..     N(I,J) .LE. 0 FOR A NODE REFERENCE
90    IF(N(I,J).GE.0) CALL ER(102,0,0)
C  SET N(I,J) TO REFLECT PROPER SUBNODE REFERENCE
      N(I,J) =N(I,J)+1-NUMBER(PCONTW)
C   N(I,J) SET TO MINUS ONE AUTOMATICALLY WHEN  NODE REF FOLLOWING AST
C   ENCOUNTERED   ,THEREFORE ONE IS ADDED TO BALANCE THIS FOR CASES WHE
C  CASES WHERE THERE FOLLOWS FURTHER SUBNODE SPECIFICATION..
      GO TO 1
10    CONTINUE
C..
C.. DELIMITER IS A LEFT PARENTHESIS FOLLOWED BY AN INTEGER
C.. FOLLOWING LOGIC ADVANCES  TO THE NEXT STACK TO BUILD THE PS ROUTINE
C.. WHICH IS ASSOCIATED WITH THE VERB IMMEDIATELY PRECEDING THE LEFT
C.. PARENTHESIS
C..
C.. LEFT PAREN SHOULD NOT FOLLOW A NODE REFERENCE
C..
      IF(N(I,J).LT.0) CALL ER(103,,0)
C.. IPSVB IS COUNT OF VERB REFERENCES WITH WHICH PS STRINGS ARE RELATED
       IPSVB = IPSVB + 1
C.. ENTER IPSVB NUMBER IN N(I,J) TO LINK VERB REFERENCE IN THIS STACK
C.. WITH LINKAGE ROUTINE IMPLEMENTING ITS PS STRINGS
       N(I,J) = N(I,J) + IPSVB
C.. COUNTER J IS SAVED FOR STACK I BEFORE STARTING NEW STACK, THIS
C.. ALLOWS RESUMPTION OF PROCESSING IN OLD STACK WHEN NEW ONE FINISHED.
      JS(I) = J
C..
```

```
C.. START NEW STACK BY INCREMENTING STACK COUNTER I
      I = I + 1
      IF (I .GT. IMAX) CALL ER (7,5,0,PCONTW)
C..
C.. ARRAY IP CONTAINS NUMBER OF P10000 ROUTINE CREATED FROM STACK I
      IP(I) = IPSVR
C..
C.. INITIALIZE POINTERS FOR NEW STACK ( J= STACK ENTRY, IPP= LOCATION
C..                                     OF ARGUMENT STRING   P=  )
      J = 1
      IPP(I,J) = 0
      N(I,J)=4096*(IPDELW-900)
C  PS ENTRY POINT NUMBER IS SAVED
      IF(PCONTW.EQ.ANULL) GO TO 101
102   S(I,J)=PCONTW
      GO TO 1
C..
C.. NULL CONTENT WORD AFTER   (I= ... , GET NEXT DELIMITER
101   CALL ADVSCA
      IF(IPDELW.GT.100) GO TO 1010
      IF(IPDELW.EQ.99)  GO TO 1009
      IF(IPDELW.EQ.IRPAR)GO TO 1011
      IF (IPDELW.NE.IAST) CALL ER(104,,0)
C.. ERROR IF A NODE REFERENCE IS NOT THE CAUSE OF NULL WORD AFTER LPAR
      N(I,J)=-N(I,J)-1
C     SET UP N(I,J) TO REFLECT SUBNODE REF.,FOR NODE REF.....
      GO TO 102
1009  S(I,J)=ANULL
      GO TO 40
1010  S(I,J)=ANULL
C.. CHECK TO INSURE LEFT PAREN NOT FOUND AFTER NULL VERB REFERENCE
C.. AND THAT PS NUMBER WITHIN RANGE.
      IF (IPDELW .GT. 120) CALL ER(104,3,0)
      GO TO 30
1011   S(I,J)=ANULL
       GO TO 60
30    CONTINUE
C..
C.. DELIMITER IS A DOLLAR SIGN FOLLOWED BY INTEGER PS STRING INDEX
C.. PS STRING SHOULD NOT OCCUR IN MAIN LEVEL OF NODE ( I .EQ. 1)
      IF(I.EQ.1) CALL ER(104,8,0)
      J=J+1
      N(I,J)=4096*(IPDELW-100)
C  ?? ABOVE TRIGGERS ENTRY POINT FOR PS SUBROUTINE??
      IPP(I,J)=0
      IF(PCONTW.EQ.ANULL) GO TO 301
302    CONTINUE
      S(I,J)=PCONTW
      GO TO 1
3009  S(I,J)=ANULL
      GO TO 40
3010  S(I,J)=ANULL
C  CHECK TO INSURE THAT LPAR NOT ISSUED AFTER ANULL CASE....
      IF (IPDELW .GT. 120) CALL ER(104,3,0)
      GO TO 30
3011  S(I,J)=ANULL
      GO TO 60
```

```
301    CALL ADVSCA
       IF(IPDELW.GT.100) GO TO 3010
       IF(IPDELW.EQ.99) GO TO 3009
       IF(IPDELW.EQ.IRPAR) GO TO 3011
       IF(IPDELW.NE.IAST) CALL ER(105,0,0)
       N(I,J)=-N(I,J)-1
       IF(PCONTW.EQ.ANULL) CALL ER(   106,,0)
        GO TO 302
C..
C.. DELIMITER IS A SLASH, SUBNODE ENTRY POINT EXPECTED
C..
C.. SUBNODE ENTRY SHOULD OCCUR ONLY AT MAIN NODE LEVEL
C..
   70 IF (I .NE. 1) GO TO 7066
7050   I=J+1
       IF (J .GT. JMAX) CALL ER (7,6,0,PCONTW)
       IPP(I,J)=0
       N(I,J)=4096*NUMBER(PCONTW)
C    ?? TRIGGER ENTRY POINT FLAG FOR NODE SUBROUTINE??
       CALL ADVSCA
       IF(IPDELW.NE.ISLASH) CALL ER(108,,0)
C  ?? ERROR IN SUB-NODE DEFINITION,,,DETECTED ON SCAN PHASE??
       IF(PCONTW.EQ.ANULL) GO TO 701
702    S(I,J)=PCONTW
       GO TO 1
701    S(I,J)=ANULL
C    ABOVE DONE IN CASE ADVSCA GOES TO NODEND??
       CALL ADVSCA
       IF(IPDELW.NE.IAST) GO TO 7055
       N(I,J)=-N(I,J)-1
        GO TO 702
7055   IF(IPDELW.NE.IDOL) GO TO 7056
       GO TO 1
7056   IF(IPDELW.NE.ISLASH) CALL ER(109,,0)
C  CALL ER ERROR IN SUBNODE DEFINITION??
       GO TO 7050
7066    CONTINUE
        IERSW=1
        I=1
       CALL ER(107,2,1)
        GO TO 7050
C..
C.. DELIMITER IS A RIGHT PARENTHESIS
C.. THIS INDICATES THE END OF A STACK SO THE CURRENT STACK IS WRITTEN
C.. OUT BY A CALL TO WRISTA AND THE PREVIOUS STACK IS REACTIVATED.
C..
60     CONTINUE
        IF(I.EQ.1) GO TO 6066
C  CHECK FOR  EXCESS OF RPARS OVER LAPARS.....
       CALL WRISTA
       I=I-1
       J=JS(I)
C.. A RIGHT PARENTHESIS SHOULD BE FOLLOWED BY A DELIMITER IMMEDIATELY
 6050 IF (PCONTW.NE.ANULL) CALL ER(111,0,0)
       GO TO 1
6066    CONTINUE
        IERSW=1
```

```
       I=1
       CALL ER(107.1,1)
       GO TO 1
C..
C.. DELIMITER IS A  (P=  COMBINATION INDICATING BEGINNING OF AN
C.. ARGUMENT STRING. STORE POINTER TO BEGINNING OF STRING IN ARRAY IPP.
C.. THEN ADVANCE SCAN PAST ARGUMENT STRING IN HOLD  TABLE.
C..
20     CONTINUE
       IPP(I,J)=ISCAN
8991   CALL ADVSCA
       IF (IPDELW .EQ. IRPAR) GO TO 6050
       IF (IPDELW.GT.100) GO TO 201
       IF(IPDELW .EQ. IDOL) GO TO 80
       IF(IPDELW .EQ. ICOMMA) GO TO 8991
       IF (IPDELW.EQ.IPER) GO TO 8991
       IF (IPDELW.EQ.IAST) GO TO 8991
       IF (IPDELW .EQ. IMINUS) GO TO 8991
       CALL ER (105.1,0)
       GO TO 8991
201    IPDELW=IPDELW+800
       GO TO   10
C   ABOVE GOES TO LPAR...
C..
C.. DELIMITER IS A  $P=  COMBINATION INDICATING BEGINNING OF AN
C.. ARGUMENT STRING AFTER A PS STRING. STORE POINTER TO BEGINNING OF
C.. STRING IN ARRAY IPP.  SINCE THIS IS AFTER A PS STRING, A NEW STACK
C.. HAS BEEN STARTED AND THESE ARGUMENTS ARE ASSOCIATED WITH THE VERB
C.. IN THE PREVIOUS STACK, THAT IS, THE VERB WHOSE PS STRINGS ARE
C.. BEING PROCESSED.
C..
40     CONTINUE
       KJ=JS(I-1)
       IPP(I-1,KJ)=ISCAN
991    CALL ADVSCA
       IF (IPDELW .EQ. IRPAR) GO TO 60
C  ABOVE GOES TO RPAR ON CONDITION
       IF (IPDELW.GT.100) GO TO  30
C  ABOVE GOES TO DOLSPLIT ON CONDITION
       IF (IPDELW.EQ.ICOMMA) GO TO 991
       IF (IPDELW.EQ.IPER  ) GO TO 991
       IF (IPDELW.EQ.IAST  ) GO TO 991
       IF (IPDELW .EQ. IMINUS) GO TO  991
       CALL ER(105.1,0,ISCAN)
       GO TO 991
80     CALL ER(110.,0)
C  ?? DOLLAR SIGN HIT...ADVSCA SHOULD HACE SENSED NODEND FIRST.....??
       GO TO 1
       END
```

D. SUBROUTINE MDSCAN

This routine initiates the model description scan at the first node and controls Phase I of the node scan. When the end of the node is encountered, routine NODEND is called which begins Phase II of the node scan. Entry point NEXNOD continues the scan of the next node of the model description at Phase I. When the last node has been processed, control is transferred to the routine MODEL which controls the fourth step in the assembly process, model program creation.

The general procedure of MDSCAN is to enter the node description into the HOLD table with a call to EXPA and then scan the node description, making lists of verbs used and expanding nonsimple verbs in the HOLD table. The list of nodes in a model description is kept in array NODNAM. The list of verbs encountered in the current node is kept in array VRBNOD and is used to calculate the data requirements for the node in routine NODFMT. The list of simple verbs encountered in the entire model description is kept in array KSIMVB. The positional data for these verbs in the library is kept in array KSIMVB. Additional verbs and routines are added to array SIMVB by the call to routine LIBCR. When a nonsimple verb is encountered, it is expanded by calls to LIBTYP and EXPA. A search is made of the verbs referenced in a nonsimple verb to prevent recursive definition where the nonsimple verb references itself. The list used for this check is kept in array VREP.

The scan through the HOLD table is accomplished by successive calls to ADVSCA which returns a delimiter code and the succeeding content word. This scan performs very little syntax error checking, the majority of it being performed in HTSCAN in Phase II.

```
      SUBROUTINE MDSCAN
C..
C.. THIS ROUTINE INITIATES THE MODEL DESCRIPTION SCAN AT THE FIRST NODE
C..
C.. ENTRY POINT NEXNOD CONTINUES THE SCAN OF THE NEXT NODE IN THE
C.. MODEL DESCRIPTION
C..
C.. EACH NODE IS HANDLED IN TWO PHASES.
C..     PHASE 1 IS CONTROLLED BY MDSCAN AND SETS UP THE HOLD TABLE
C..             THROUGH CALLS TO EXPA.
C..     PHASE 2 IS CONTROLLED BY HTSCAN WHICH SCANS THE HOLD TABLE
C..             AND CONSTRUCTS THE REQUIRED LINKAGE ROUTINES.
C..
C.. WHEN THE LAST NODE HAS BEEN PROCESSED, CONTROL IS TRANSFERED TO
C.. THE ROUTINE MODEL WHICH CONTROLS THE SELECTION OF MODULES FROM
C.. INTERNAL FILES, THE SELECTION AND DIMENSIONING OF COMMON DECKS,
C.. AND THE CONSTRUCTION OF A COMPLETE MODEL TAPE.
C..
C..
      COMMON /GCHARC/ NDEL,NEND,STRING(1),BUFF(80),LUNIT,INC,ICD,LOUT
     1               ,OLDBUF(80),LOCATE(10),CDSCAN,NEWCRD,NCHAR
      COMMON  /VLIB/  IV,VI(600),ISI(600),NPSLOT(600),NVMAX
      COMMON /SIMVB / SIMVB(400),VREP(400),ISIM,KSIMVB(400),INDVB,
     1                INDLIB,NSVMAX,VRBNOD(300),NVNOD,NVNMAX
      COMMON /SWITCH/ IXFL,LIBFL,KEYFMT,LSTMOD,ISTDSR,MUPDAT
      COMMON /COMDEK/ COMNAM(60),INDNAM,NAMMAX,
     1                COMDIM(80,2),INDDIM,MAXDIM,
     2                NODNAM(100),INDNOD,NODMAX,
     3                INITCD(60),LASTCD(60)
      COMMON /ADDIT / IRET
      COMMON /ADVSCA/ ISCAN,INEND,IPDELM,PCONTW,IVR
      COMMON /DTIM  / KC,DTIM
      COMMON /ER    / ICDH,ANM1
      COMMON /EXPA  / IDOL,ICOMMA,LPAR,IEQUAL,IRPAR,IPER,IAST,ISLASH,
     1                IPLUS,IMINUS
      COMMON /KN    / KNODE
      COMMON /MAIN  / AVB,IFREEH,IENDS,ANWND,ANULL
      COMMON    HOLD(2000)
      EQUIVALENCE    (IHOLD(1),HOLD(1))
      DIMENSION       IHOLD(1)
      DIMENSION       ICAPPT(100)
      DIMENSION       BUFH(80)
      DATA            AMSK  /07777777777555555555/
      EQUIVALENCE (ANWND,NEWNOD)
C..
C.. INITIATE NODE SCAN, SET VARIABLES FOR CALL TO GCHAR
C..
      LUNIT = 5
   60 ICD = 0
   61 LOUT = 6
      CALL GCHAR
      IF (NEND .NE. 0) GO TO 64
C.. CHECK FOR INITIAL COMMENT CARD
      IF (NDEL .NE. IPLUS) CALL ER(1.0,0,ANWND)
      INC = 200
      GO TO 61
C.. A NEW NODE NAME IS ASSUMED
```

```
   64 ANAND = STRING(1)
      IF (NDEL .NE. IPER) CALL ER(1,0,0,ANWND)
C.. THE FIRST DELIMITER AFTER A NODE NAME WAS NOT A PERIOD
   65 CONTINUE
C.. ENTER NODE NAME IN LIST
      DO 70 K=1,INDNOD
      IF (NEWNOD .EQ. NODNAM(K)) GO TO 72
   70 CONTINUE
      INDNOD = INDNOD + 1
      IF (INDNOD .GT. NODMAX) GO TO 71
      NODNAM(INDNOD) = NEWNOD
      GO TO 74
   71 CALL ER(31,0,1,ANAND)
      GO TO 74
   72 CALL ER(32,0,1,ANAND)
C.. CLEAR LIST OF VERBS ENCOUNTERED IN CURRENT NODE
   74 CALL CLEAR (VRBNOD,NVNMAX)
      NVNOD = 0
      ISCAN=0
      IVR=0
      ICAP=0
C.. PLACE ONE NODE INTO HOLD( )
      KNODE=KNODE + 1
      CALL EXPA(0)
      ICDH=ICD
      INCH=INC
      DO 75 K=1,80
   75 BUFH(K)=BUFF(K)
C.. BEGIN PROCESSING NODE DESCRIPTION
   80 CONTINUE
C..   PLACE NEXT VERB IN AVB
   85 CALL ADVSCA
      I160=0
   89 IF (IPDELW.EQ.99 .OR. IPDELW.EQ.899) GO TO 165
   90 IF( IPDELW.EQ.IAST ) GO TO 155
      IF( IPDELW.EQ.ISLASH)GO TO 160
      IF( IPDELW.EQ.IDOL)  CALL NODEND
C.. MASK OUT RIGHTMOST FIVE CHARACTERS
   95 AVB=PCONTW.AND.AMSK
      IF(AVB.EQ.ANULL) GO TO 85
C.. CHECK IF ONLY STRUCTURE EXPANSION
   97 IF(IXFL.EQ.1) GO TO 80
C..
C..
C..       ADD TO LIST OF  ALL   VERBS IN THIS NODE  (VRBNOD) IF
C..       NOT PREVIOUSLY LISTED.
C..
      DO 98 K=1,NVNOD
         IF (AVB .EQ. VRBNOD(K)) GO TO 99
   98 CONTINUE
      NVNOD = NVNOD + 1
      IF (NVNOD .GT. NVNMAX) CALL ER (9,0,0,AVB)
      VRBNOD(NVNOD) = AVB
   99 CONTINUE
C.. IS VERB SIMPLE OR NON-SIMPLE
      CALL LIBTYP(ISF)
      IF (ISF .EQ. 0) GO TO 110
```

```
      IF (ISF .GE. 2) CALL ER(16,0,1,AVB)
C..
C.. VERB IS SIMPLE,STORE IN ARRAY SIMVB IF NOT ALREADY THERE
C..
      IF(ISIM.EQ.0) GO TO 105
      DO 100 K=1,ISIM
         IF(AVB.EQ.SIMVB(K)) GO TO 80
  100 CONTINUE
  105 ISIM=ISIM+1
      IF(ISIM.GT.NSVMAX) CALL ER(7,0,0,AVB)
      SIMVB(ISIM)=AVB
      KSIMVB(ISIM)=INDVB + 4096*INDLIB
C..
C.. CHECK CROSS REFERENCE FILE TO INSURE THAT ALL REFERENCED MODULES
C.. ARE INCLUDED IN THE LIST SIMVB
C..
      CALL LIRCR(AVB)
      GO TO 80
C.. VERB IS NON-SIMPLE
C..   EXPA ALTERS RIGHTMOST CHARACTER OF PCONTW IF VERB NAME WAS
C..   EMBEDDED WITHIN THE  N-S  VERB DEFINITION. A N-S VERB CANNOT
C..   REFERENCE ITSELF, THEREFORE ALL VERB NAMES IN A N-S VERB
C..   DEFINITION MUST BE CHECKED AGAINST N-S VERB NAME FOR POSSIBLE
C..   RECURSIVE DEFINITIONS
C..
  110 CONTINUE
      IF(IVR.EQ.0) GO TO 145
      IF(AVB.NE.PCONTW)  GO TO 140
  115 CONTINUE
      IF(ICAP.EQ.0) GO TO 135
      ICAPL=ICAPPT(ICAP)
      IF(IVR.LT.ICAPL) GO TO 130
  120 CONTINUE
C.. ICAPL IS TOP LIMIT OF SCAN FOR RECURSIVE DEFINITION
C.. IVR   IS LIMIT OF VREP STACK
C..
      DO 125 K5=ICAPL,IVR
         IF(AVB.EQ.VREP(K5)) CALL ER(2,0,0,AVB)
  125 CONTINUE
      GO TO 145
  130 CONTINUE
      ICAP=ICAP - 1
      GO TO 115
  135 CONTINUE
      ICAPL=1
      GO TO 120
C..
C.. LIMIT SEARCH OF VERB NAMES TO AVOID CONFUSING VERBS IN PARAMETER
C..   SLOTS WITH THOSE IN REGULAR FLOW PATH.
C..
  140 CONTINUE
      IVR=IVR + 1
      IF(IVR.GT.100) CALL ER(2,5,0,AVB)
      VREP(IVR)=AVB
      ICAP=ICAP + 1
      ICAPPT(ICAP)=IVR
      GO TO 150
```

```
  145 IVR=IVR + 1
      IF(IVR.GT.100) CALL ER(2,6,0,AVR)
      VREP(IVR)=AVR
C.. EXPAND VERB
  150 CONTINUE
      CALL TABLE
      CALL EXPA(1)
      IF(I160.NE.1) GO TO 80
      CALL ADVSCA
      I160=0
      GO TO 95
  155 CALL ADVSCA
      I160=0
      IF(IPDELX.EQ.IPER) GO TO 85
      GO TO 89
  160 CALL ADVSCA
      I160=1
      GO TO 95
  165 CALL ADVSCA
      I160=0
      IF (IPDELX.EQ.IRPAR.OR.IPDELX.GT.100) GO TO 90
      GO TO 165
C..
C.. THIS ENTRY POINT IS FOR RETURN FROM  HTSCAN  SUBROUTINE
      ENTRY NEXNOD
C.. IF LAST NODE HAS BEEN PROCESSED, TRANSFER CONTROL TO  *MODEL*
      IF (IENDS .EQ. 1) CALL MODEL
      LOUT=6
      LUNIT=5
      ICD=ICDH
      INC=INCH
      DO 170 K=1,80
  170    BUFF(K)=BUFH(K)
      WRITE (6,2170) ICD,BUFF
 2170 FORMAT(1H1,I4,2H..,80A1)
      GO TO 65
      END
```

E. SUBROUTINE NODEND

This routine is called by routine ADVSCA when the end of a node is encountered in the HOLD table. If Phase I (IFL = 1) of the node scan is in progress, this signals the end of Phase I and IFL is set to 2 and the routine HTSCAN is called to initiate Phase II. If Phase II is in progress when NODEND is called, this signals the end of the node scan and the node subroutine is written out from the first level stack by a call to routine WRISTA. Routine NODFMT is then called if input data requirements are requested. The variable IENDS is set to 1 by routine EXPA when the final node in the model description is loaded in the HOLD table. Therefore at the end of routine NODEND, the variable IENDS is checked and routine MODEL is called if its value is 1. If its value is not equal to 1, the entry point NEXNOD of routine MDSCAN is called which initiates Phase I of the scan of the next node in the model description.

```
      SUBROUTINE NODEND
      COMMON /IERCT/ IERCT
      COMMON /I/    I,J
      COMMON     /APHAS/  IFL
      DATA IFL/1/
      COMMON /ADDIT/    IRET
      COMMON /ADVSCA/ ISCAN,INEND,IPDELM,PCONTM,IVR
      COMMON /ER/ ICOM,ANM1
      COMMON /EXPA/ IDOL,ICOMMA,LPAR,IEQUAL,IRPAR,IPER,IAST,ISLASH,IPLU
     1S,IMINUS
      COMMON /GCHARC/ NDEL,NEND,STRING(1),BUFF(80),LUNIT,INC,ICD,LOUT
     1                ,OLDBUF(80),LOCATE(10),CDSCAN,NEWCRD,NCHAR
      COMMON  /VLIB/  IV,VI(600),ISI(600),NPSLOT(600),NVMAX
      COMMON /MAIN/ AVR,IFREEH,IENDS,ANAND,ANULL
      COMMON /SWITCH/ IXFL,LIBFL,KEYFMT,LSTMOD,ISTOSR,MUPDAT
      COMMON /MHOLD/ MAXHLD
      COMMON HOLD(1)
      DIMENSION IHOLD(1)
      EQUIVALENCE (HOLD(1),IHOLD(1))
      DATA ISAVER  /0/
      DATA  ICHECK /16000000/
C..
C.. THIS ROUTINE IS CALLED FROM ADVSCA WHEN THE END OF THE NODE BEING
C.. PROCESSED IS ENCOUNTERED. THIS OCCURS TWICE FOR EACH NODE --
C.. DURING PHASE I (IFL=1) WHEN THE NODE IS BEING EXPANDED  AND
C.. DURING PHASE II (IFL=2) WHEN CODE IS BEING GENERATED FOR LINKAGE
C.. ROUTINES.
C..
      GO TO (1,2),IFL
C.. PHASE I, CALL HTSCAN TO START PHASE II PROCESSING
C.. NO RETURN IS EXPECTED FROM HTSCAN.
1     IFL=2
      CALL HTSCAN
C..
C.. PHASE II, CHECK THAT STACK LEVEL 1 IS BEING PROCESSED.
C..
2     IFL=1
       IF(I.NE.1) GO TO 66
C    AT END OF NODE FINAL WRISTA MUST BE FIRED TO CLEAR OUT NODE CODE
      CALL WRISTA
67     CONTINUE
C.. WRITE OUT DATA FORMATS FOR THIS NODE ON FILE 9
      IF (KEYFMT .EQ. 1) CALL NODFMT
C..
C..  IF ERRORS IN THIS NODE, PRINT OUT THE CONTENTS OF THE HOLDTABLE
C..
      IF (ISAVER .GE. IERCT) GO TO 20
      ISAVER = IERCT
      K=-1
      WRITE(6,100)
  100 FORMAT(1H0,5X,11HHOLDTABLE    //)
   10 CONTINUE
      K = K + 2
      KP = K + 1
C..  CHECK FOR OVERFLOW OF HOLDTABLE
      IF (K .GT. MAXHLD .OR. K.GE.IFREEH) GO TO 15
C..  CHECK FOR CONTENT WORD OR INTEGER CODE
```

```
      IPRSWT = 1
      IF (IHOLD(K).LT.0 .AND. IHOLD(K).GT.-ICHECK .OR.
     *    IHOLD(K).LT. ICHECK .AND. IHOLD(K).GT.0) IPRSWT = 3
      IF (IHOLD(KP).LT.0 .AND. IHOLD(KP).GT.-ICHECK .OR.
     *    IHOLD(KP).LT.ICHECK .AND. IHOLD(KP).GT.0) IPRSWT=IPRSWT+1
C..   SELECT PROPER PRINT FORMAT
      GO TO (11,12,13,14), IPRSWT
   11 WRITE(6,111) K,HOLD(K),KP,HOLD(KP)
  111 FORMAT(1X,I4,3H...,A10,5X,I4,3H...,A10)
   12 WRITE(6,112) K,HOLD(K),KP,IHOLD(KP)
  112 FORMAT(1X,I4,3H...,A10,5X,I4,3H...,I5)
      GO TO 10
   13 WRITE(6,113) K,IHOLD(K),KP,HOLD(KP)
  113 FORMAT(1X,I4,3H...,I5,10X,I4,3H...,A10)
      GO TO 10
   14 WRITE(6,114) K,IHOLD(K),KP,IHOLD(KP)
  114 FORMAT(1X,I4,3H...,I5,10X,I4,3H...,I5)
      GO TO 10
   15 WRITE(6,115)
  115 FORMAT(//1X,24H,,, END OF HOLDTABLE ***   )
C..
C.. CALL MODEL IF THIS IS FINAL NODE IN MODEL DESCRIPTION OR
C.. CALL NEXNODE ENTRY POINT IN MDSCAN TO PROCESS NEXT NODE.
C.. RETURN IS NOT EXPECTED FROM EITHER ROUTINE.
C..
   20 IF (IENDS .EQ. 1) CALL MODEL
      CALL NEXNOD
66     CONTINUE
      CALL ER(107,3,1)
      GO TO 67
      END
```

F. SUBROUTINE NODFMT

This routine selects the format and data requirements for each node from file 4 and writes them out with headings on file 9 for later printing by routine MODFMT. Routine NODFMT is called at the end of each node by routine NODEND and uses the list of modules in array VRBNOD to determine which modules to retrieve the data requirements from. This routine is also called from the main program AUTASM. This first call to NODFMT initializes file 9 with a header page. If the standard subroutine package is selected, the routine prints out all the data requirements from the module DATAN but none from any other routine in the standard package.

A node header line with the node name is printed out first. A module name is printed out prior to the contents of all C/F cards in the module. Only columns 5-72 of each C/F card are used. If a module listed in array VRBNOD contains no C/F cards, the module name is not printed. If a node contains no modules with C/F cards, the node header line is not printed.

```
      SUBROUTINE NODFMT
C..
C.. THIS ROUTINE SELECTS THE FORMAT AND DATA REQUIREMENTS FOR EACH NODE
C.. FROM FILE 4 AND WRITES THEM OUT WITH HEADINGS ON FILE 9 FOR LATER
C.. PRINTING. NODFMT IS CALLED AT THE END OF EACH NODE AND USES THE
C.. LIST OF MODULES IN ARRAY VRBNOD.
C..
      COMMON /GCHARC/ NDEL,NFND,STRING(1),BUFF(80),LUNIT,INC,ICD,LOUT
     1                ,OLDBUF(80),LOCATE(10),CDSCAN,NEWCRD,NCHAR
      COMMON /MAIN  / AVB,IFREEH,IENDS,ANWND,ANULL
      COMMON /SIMVB / SIMVB(400),VREP(400),ISIM,KSIMVB(400),INDVB,
     1                INDLIB,NSVMAX,VRBNOD(300),NVNOD,NVNMAX
      COMMON /SWITCH/ IXFL,LIBFL,KEYFMT,LSTMOD,ISTDSR,MUPDAT
      COMMON  /KN   / KNODE
      COMMON /COMDEK/ COMNAM(60),INDNAM,NAMMAX,
     1                COMDIM(80,2),INDDIM,MAXDIM,
     2                NODNAM(100),INDNOD,NODMAX,
     3                INITCD(60),LASTCD(60)
      DIMENSION BUFH(15)
      EQUIVALENCE (IVRBNM,VRBNAM), (ITSTNM,TSTNAM)
      DATA     INITL   /0/
      DATA      MODEL  /5HMODEL/
      DATA     ADATAN  /5HDATAN/
      DATA     I7      /1H7    /
      DATA     I9      /1H9    /
      NPAGE = 0
      INDEX = 0
      LINE  = 0
       NODE = NODNAM(KNODE)
C..
C.. INITIALIZE FILE 9
C..
      IF (INITL .EQ. 1) GO TO 10
      INITL = 1
C.. WRITE HEADER PAGE
      WRITE (9,2000)
 2000 FORMAT (1H1////////////////////////////
     1        47X,36H***   MODEL DATA REQUIREMENTS   *** )
C.. CHECK IF STANDARD SUBROUTINE PACKAGE IS TO BE USED
      IF (ISTDSR .EQ. 1) RETURN
       NODE =  MODEL
      VRBNOD(1) = ADATAN
      NVNOD = 1
C..
C.. START PROCESSING ALL MODULES IN VRBNOD FOR NODE
   10 NODHDR = 0
C.. GET NEXT MODULE NAME
   20 INDEX = INDEX + 1
      IF (INDEX .GT. NVNOD ) RETURN
      VRBNAM = VRBNOD(INDEX)
C..
C.. FIND MODULE WITH NAME VRBNAM ON DATA FORMATS FILE 4
   29 NOTLST = 0
      IFIRST = 1
      NAMHDR = 0
   30 READ  (4,1030) IKEY,TSTNAM,TYPE
 1030 FORMAT(A1,A6,A6)
```

```
      NOTLST = NOTLST + 1
      IF (EOF,4) 40,50
C.. END OF FILE, MODULE NOT FOUND
   40 REWIND 4
      IF (NOTLST .LE. 1) GO TO 30
   45 WRITE  (6,2040) VRBNAM
 2040 FORMAT (14H MODULE NAMED ,A6,31H NOT FOUND, DATA FORMATS FILE 4 )
      GO TO 20
C.. CHECK MODULE NAME
   50 IF (IKEY .NE. I9) GO TO 30
      IF (IVRBNM - ITSTNM) 60,80,70
C.. VRBNAM ON PREVIOUS PORTION OF FILE
   60 REWIND 4
      IF (IFIRST .NE. 1) GO TO 45
      IFIRST = 0
      GO TO 30
C.. VRBNAM FORWARD ON FILE, SKIP THIS MODULE
   70 READ (4,1030) IKEY
      IF (IKEY .EQ. I7) GO TO 30
      GO TO 70
C.. VRBNAM FOUND, READ CONTENT FROM FILE 4
   80 READ (4,1080) IKEY,(BUFH(I),I=1,13)
 1080 FORMAT(A1,A4,11A6,A2)
      IF (IKEY .NE. I7) GO TO 90
C.. END OF CONTENT
      GO TO 20
C..
C.. PRINT NODE HEADER IF NECESSARY
C..
   90 IF (NODHDR .EQ. 1) GO TO 92
      NODHDR = 1
      NPAGE = NPAGE + 1
      WRITE (9,2010)  NODE,NPAGE
 2010 FORMAT (1H1,50X,24HDATA REQUIREMENTS, NODE ,A6,40X,4HPAGE,I2,///)
C.. PRINT MODULE HEADER
   92 IF (NAMHDR .EQ. 1 ) GO TO 95
      WRITE (9,2020) VRBNAM
 2020 FORMAT (1H0,10X,7HMODULE ,A6 /1H0 )
      NAMHDR = 1
      LINE = LINE + 4
C.. WRITE COLUMNS  5 - 72 ONTO FILE 9
   95 WRITE (9,2095) (BUFH(I),I=2,13)
 2095 FORMAT(29X,12A6)
      LINE = LINE + 1
      IF (LINE .LE. 50) GO TO 80
      LINE = 0
      NPAGE = NPAGE + 1
      WRITE (9,2010)  NODE,NPAGE
      GO TO 80
      END
```

G. SUBROUTINE PSTRIN(K)

This routine sets up a table of arguments for a FORTRAN subroutine CALL of a verb routine. The argument K indicates which verb in the current level stack to build the table of arguments for. The array IPP(I,K) points to the beginning of the argument string in the HOLD table for the verb. The array WR contains the characters for the argument string with the content words in the odd positions and the delimiter characters in the event position. The delimiter characters are period, comma, minus, and asterisk. An asterisk is changed to a 5H for the hollerith argument. The final character in the table is always a right parenthesis. The beginning left parenthesis is assumed and therefore is not entered in the table. The variable KPWRIT points to the position of the right parenthesis in the table. As an example, for the call to a verb

CALL VVERB (1,3.1, *ANAME) the array WR would contain

WR(1) = 1	WR(6) = ,
WR(2) = ,	WR(7) = blank
WR(3) = 3	WR(8) = *
WR(4) = .	WR(9) = ANAME
WR(5) = 1	WR(10) =)

and the value of KPWRIT would be 6. The delimiter codes in the HOLD table are translated to delimiter characters through the array DEL.

This routine is called by routine WRISTA prior to the call of routine PACK1 which creates the call to the verb and uses the table of arguments in WR.

```
      SUBROUTINE PSTRIN(K)
C..
C.. THIS ROUTINE SETS UP A TABLE OF ARGUMENTS FOR A FORTRAN SUBROUTINE
C.. CALL OF A VERB ROUTINE
C..
      COMMON  HOLD(1)
      COMMON /WRT/  KPWRIT,WR(40),KPMAX
      COMMON /I/    I
      COMMON /SCAN/ S(20,50),IPP(20,50),N(20,50),IP(20),JS(20)
     1              ,IMAX,JMAX
      COMMON /ADDIT/   IRET
      COMMON /ADVSCA/ ISCAN,INEND,IPDELW,PCONTW,IVR
      COMMON /ER/ ICOM,ANM1
      COMMON /EXPA/ IDOL,ICOMMA,LPAR,IEQUAL,IRPAR,IPER,IAST,ISLASH,IPLU
     1S,IMINUS
      COMMON /GCHARC/ NDEL,NEND,STRING(1),BUFF(80),LUNIT,INC,ICD,LOUT
     1               ,OLDBUF(80),LOCATE(10),CDSCAN,NEWCRD,NCHAR
      COMMON  /VLIB/  IV,VI(600),ISI(600),NPSLOT(600),NVMAX
      COMMON /MAIN/ AVR,IFREEH,IENDS,ANWND,ANULL
      EQUIVALENCE   (IHOLD(1),HOLD(1))
      DIMENSION  IHOLD(1)
      DATA   AMSK   /77777777775555555558/
      DIMENSION DEL(10)
      DATA  NDELCT,(DEL(K),K=1,10)/10,1H$,1H,,1H(,1H=,1H),1H,,1H*,1H/,1
     1H+,1H-/
      INTEGER RPAR
      EQUIVALENCE (RPAR,IRPAR)
      DATA  BLNK   /1H /
      DATA  ASH   /2H5H/
C     SET ISCAN TO FIRST  P-VALUE  FOR THIS VERB REFERENCE
      ISCAN=IPP(I,K)
      WR(1) = HOLD(ISCAN) .AND. AMSK
      IF (WR(1) .EQ. ANULL)  WR(1) = BLNK
C  BUILD  TABLE FOR WRITING OUT   PROPER FORTRAN SUBROUTINE CALL PARAMET
C  PARAMETERS......
      IK=1
C..  ADVANCE SCAN THROUGH PARAMETER VALUES
    1 ISCAN = ISCAN + 2
      IF ((ISCAN-1) .EQ. INEND)  CALL ER(113,0,1,ISCAN)
      IPDELW = IHOLD(ISCAN-1)
      PCONTW = HOLD(ISCAN)
      PCONTW = PCONTW .AND. AMSK
C.. CHECK FOR END OF PARAMETER VALUES STRING
      IF(IPDELW.EQ.RPAR.OR.IPDELW.GT.100) GO TO 60
      IK=IK+2
      IF (IK .GT. KPMAX) CALL ER(113,5,0,ISCAN)
      IF(PCONTW.EQ.ANULL)   PCONTW=BLNK
      WR(IK)=PCONTW
      WR(IK-1)=DEL(IPDELW)
C..CHECK FOR HOLLERITH ARGUMENT AND ENTER  5H  IN PLACE OF ASTERISK
C..DELIMITER PRECEDING NAME
      IF (IPDELW .NE. IAST) GO TO 1
      WR(IK-1) = ASH
C.. INSURE THAT HOLLERITH ARGUMENT WONT SPAN TWO LINES IN THE CALL
      IF (MOD(IK-1,12) .GT. 0) GO TO 1
C.. MOVE HOLLERITH 5H DELIMITER TO NEXT LINE
      IK = IK + 2
```

```
      IF (IK .GT. KPMAX) CALL ER(113,5,0,ISCAN)
      WR(IK) = PCONTW
      WR(IK-1) = ASH
      WR(IK-2) = BLNK
      WR(IK-3) = BLNK
      GO TO 1
60    KPWRIT=IK+1
      WR(KPWRIT)= DEL(IRPAR)
      RETURN
      END
```

CHAPTER IV
OUTPUT DATA POSTPROCESSOR PROGRAM CHANGES

A. INTRODUCTION

The MAWLOGS Output Data Postprocessor System (ODPS) was developed to provide a post-model run analysis capability. This system operates on a transaction file, developed by a simulation model which contains statistical data on events and values occurring during the model run. Various ODPS programs can then be used to aggregate, analyze, and graph the results of the simulation. The use of the ODPS programs is described in the User's Manual[2] and a technical description is given in the Programmer's Guide[5].

The ODPS system has been extended to handle the method in which statistics are collected in continuous flow models. In addition, the graphical output now provides a reference to the tabular reports that back-up each variable on a graph. These extensions have affected seven programs, namely AGATE, TRAFAN, PGRAPH, ENDB, PRNTSO, REPTTL, and GRAPH. Changes to each of these programs are described and program listings are provided in this chapter. Two new programs have been provided to handle model transaction files, called MTTF2 and CHTFH. MTTF2 is used to merge two transaction files which are produced by two different models or slices of the same model. CHTFH is used to change the transaction file header identification table. Both of these new programs are documented in this chapter.

The statistics collection in continuous flow models is performed at regular, pre-specified intervals on all variables in the system. In contrast, the discrete-event modules only collect statistics when a change occurs in a variable of interest. The original ODPS programs assumed that there was a value change between two TMST type statistic entries at successive times on the transaction file and used this information in calculating these

---

[5] Fuelling, C. P., Models of the U.S. Army Worldwide Logistics System (MAWLOGS), Volume IV, Programmer's Guide, Part 3 - Technical Description of the ODPS Programs, General Research Corporation, OAD-CR-41, June 1974.

statistics. Since this assumption was not true for continuous flow model statistics, alterations had to be made in the way values were applied across time periods. These changes were made in the two main programs that read transaction files, AGATE and TRAFAN.

The output of the GRAPH package in the ODPS system has been improved in two ways. An example of a graph produced by the ODPS is shown in Figure IV-1. Note that each cross variable includes the number of the tabular report from which the graph points are derived. The last entry in each report, which contains the summary of all earlier entries, has been suppressed in the graph. In the earlier version of this program, this cumulative value caused the resolution on the vertical axis to be compressed just to handle that point.

B. MTTF2 PROGRAM

1. Purpose

This program merges two MAWLOGS model transaction files which have been sorted into ascending order on words one and two or descending order on word two. This program differs from MTTF in that the files need not be from a save run and a restart run of the same model. In MTTF, the name tables at the head of the files are assumed to be the same, or the first one a subset of the second, so that the indices on the transaction records are unique. MTTF2 does not make that assumption and accordingly offsets the indices of the second transaction file to maintain their uniqueness. MTTF2 is used when merging transaction files from different slices of the same model or from different models. In this case, one file would have transactions indexed 1 through n and the other file would be indexed 1 through m, but the same indices would not relate to the same variables. The output file from MTTF2 would have n+m transaction indices that would uniquely identify all the records on the file. A value to increment the node numbers on the second file can also be used to differentiate between the transactions from the two files.

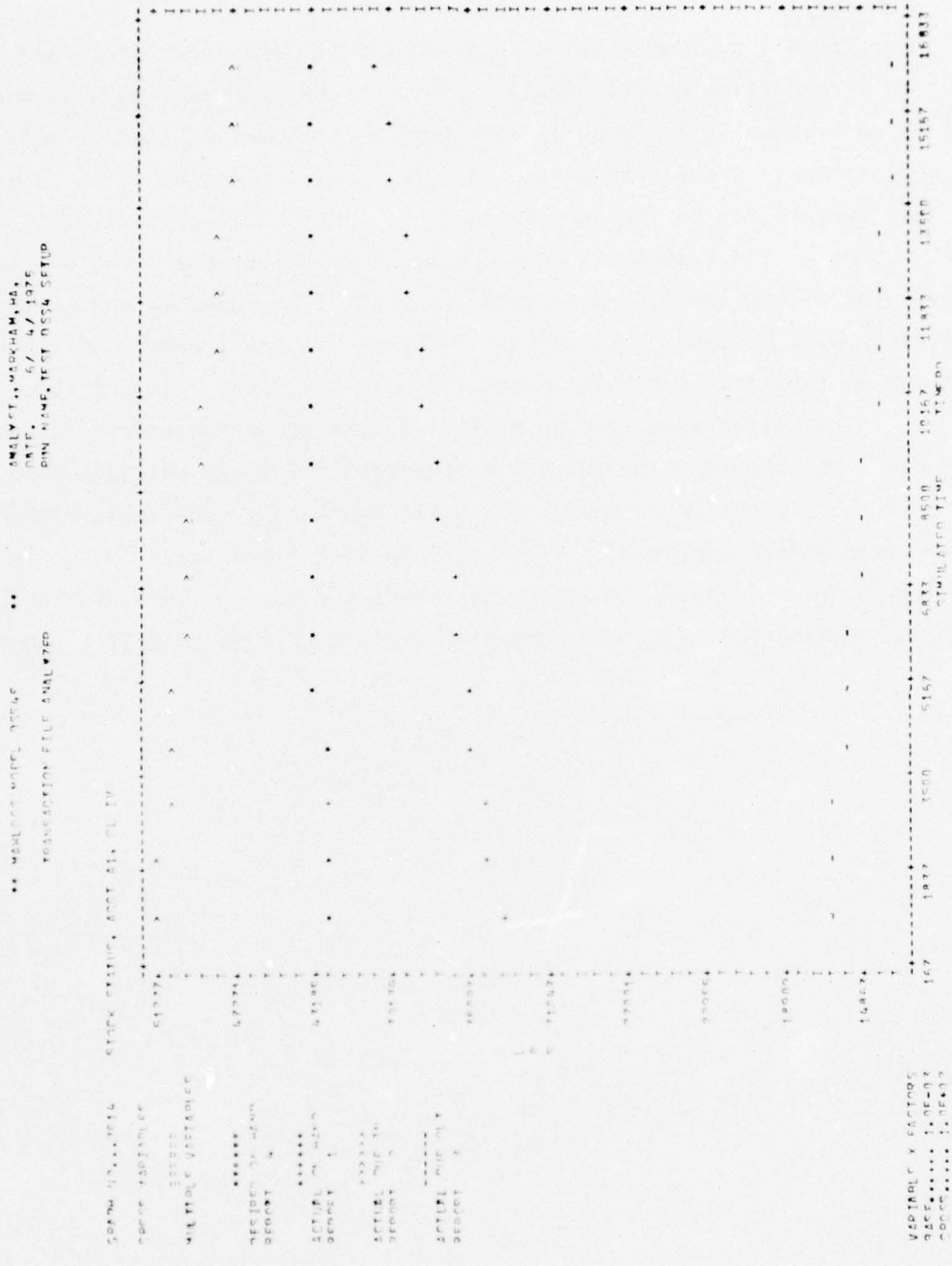


Figure IV-1. Example of an ODPS Graph

2. Use

The MTTF2 program reads from two files, TAPE1 and TAPE2, and outputs the merged files on file TAPE3. The program reads two input data cards with formats shown in Table IV-1. The node code offset is used to alter the node numbers in the transaction name table of the second file. Since the node numbers may be the same for the two files, this offsetting of the node numbers in the transaction names guarantees uniqueness among all the names. The second input card is used to supply a new name to the merged file which will be used in subsequent ODPS reports and graphs. If the name is blank on the card, the name associated with the first file will be used.

The output tape from an MTTF2 run must be sorted before it is used since the records from the two transaction files are not sorted by MTTF2 but merely placed consecutively after the merged name table records. If the same model node created statistics on both files, then the sorted file should be run through AGATE to aggregate the records from that node to a single transaction index and properly handle TMST type statistic records.

TABLE IV-1. FORMAT OF MTTF2 CARDS

CARD COLUMN	FORMAT	DESCRIPTION
1 - 5	A5	ENTER "MTTF2"
6 -10	5X	BLANK
11-20	F10.0	NODE CODE OFFSET VALUE FOR TAPE2 TRANSACTION NAMES
21-80	60X	BLANK
1 -10	10X	ENTER "RUNNAME" AS CARD TITLE
11-28	3A6	NEW RUN NAME TO HEADER OF OUTPUT FILE

```
      PROGRAM MTTF2(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,
     *              TAPE1,TAPE2,TAPE3)
C
C  THIS PROGRAM MERGES TWO MAWLOGS MODEL TRANSACTION FILES WHICH HAVE
C  BEEN SORTED INTO ASCENDING ORDER ON WORDS ONE AND TWO OR DESCENDING
C  ORDER ON WORD TWO. THIS PROGRAM DIFFERS FROM MTTF IN THAT THE FILES
C  NEED NOT BE FROM A SAVE RUN AND A RESTART RUN OF THE SAME MODEL,
C  THAT IS, THE NAME TABLES ARE NOT CONSIDERED TO BE THE SAME OR ONE
C  A SUBSET OF THE OTHER.
C
C  INPUT     TAPE1    FIRST FILE
C            TAPE2    SECOND FILE
C
C  OUTPUT    TAPE3    MERGED FILE
C
C     TAPE3 IS FORMED BY MERGING THE NAME TABLE AS READ FROM TAPE1
C  WITH THE NAME TABLE FROM TAPE2 WITH THE TRANSACTION INDICES OFFSET
C  TO FOLLOW THOSE OF TAPE1 IN ORDER. A NODE CODE OFFSET, READ FROM
C  A CARD, IS ALSO APPLIED TO MAKE THE NAMES UNIQUE FROM THOSE OF TAPE1.
C
C     THE OUTPUT TAPE SHOULD BE RUN THROUGH AGATE TO PROPERLY HANDLE
C  SUBCODE TYPE 1 STATISTICS. TAPE3 MUST BE SORTED BEFORE THE AGATE RUN.
C
C     TBEG AND TFIN ARE SELECTED TO COVER THE RANGE OF TAPE1 AND TAPE2.
C
C     INPUT CARD -
C         CC1-CC5  AS MTTF2
C         CC6-CC10 5X BLANK
C        CC11-CC20 F10.0  NODE CODE OFFSET FOR TAPE2 TCTNAMS
C
C
C         CC1 -CC7  10X RUNNAME  (CARD TITLE)
C         CC11-CC28 3A6 NEW RUN NAME
C
C
      COMMON /GSPDUM/ TBEG,TFIN,NCLCT,MON,NDAY,NYR,RUNNAM(3),
     *                MODNAM(2),NAME(2),NMDPP(5)
      DIMENSION A1(4),B1(500),A2(4),B2(500),BD(500),IA1(4),IA2(4)
      EQUIVALENCE  (IA1,A1), (IA2,A2)
      DATA  NB,NA,NFI1,NFI2,NFO,NRO,NRI1,NRI2,MODE
     *     /500,4,   1,   2,  3,  0,   0,   0,   0 /
      DATA BLANK /1H /
C
C OPEN FILES
C
      CALL INITB(B1,NB,IB1,NFI1,MODE,IEOF1)
      CALL INITB(B2,NB,IB2,NFI2,MODE,IEOF2)
      MODE = 1
      CALL INITB(BO,NB,IBO,NFO ,MODE,IEOFO)
C READ FILE2 NODE OFFSET CODE
C
      READ(5,1000) OFFSET
 1000 FORMAT(10X,F10.0)
      READ (5,1010) RUNNAM
 1010 FORMAT(10X,3A6)
C
C
```

```
C  READ FIRST SIX HEADER RECORDS FROM NFI1  AND COPY ON NFIO UNCHANGED.
C
      DO 10 I=1,6
      CALL DBLK(A1,B1,NA,NB,IB1,NFI1,IEOF1)
      IF (I.LT.5) GO TO 8
      IF (RUNNAM(1) .EQ. BLANK) GO TO 8
      IF (I.EQ.6) GO TO 6
    5 A1(4) = RUNNAM(1)
      GO TO 8
    6 A1(3) = RUNNAM(2)
      A1(4) = RUNNAM(3)
    8 CONTINUE
      CALL BLOCK(A1,BO,NA,NB,IBO,NFO)
   10 CONTINUE
C
C  SKIP FIRST SIX HEADER RECORDS FROM NFI2.
C
      DO 20 I=1,6
      CALL DBLK(A2,B2,NA,NB,IB2,NFI2,IEOF2)
   20 CONTINUE
C
C  DETERMINE TBEG AND TFIN FROM RECORD 7
C
      CALL DBLK(A1,B1,NA,NB,IB1,NFI1,IEOF1)
      TBEG1 = A1(3)
      TFIN1 = A1(4)
      CALL DBLK(A2,B2,NA,NB,IB2,NFI2,IEOF2)
      TBEG2 = A2(3)
      TFIN2 = A2(4)
      TBEG = TBEG1
      IF (TBEG .GT. TBEG2) TBEG = TBEG2
      TFIN = TFIN1
      IF (TFIN .LT. TFIN2) TFIN = TFIN2
      A1(3) = TBEG
      A1(4) = TFIN
      CALL BLOCK(A1,BO,NA,NB,IBO,NFO)
C
C  DETERMINE NTNO FROM RECORD 8
C
      CALL DBLK(A1,B1,NA,NB,IB1,NFI1,IEOF1)
      NTN1 = IA1(3)
      CALL DBLK(A2,B2,NA,NB,IB2,NFI2,IEOF2)
      NTN2 = IA2(3)
      NTNO = NTN1 + NTN2
      IA1(3) = NTNO
      CALL BLOCK(A1,BO,NA,NB,IBO,NFO)
C
C  COPY NAME RECORDS FROM NFI1 TO NFO
C
      N = NTN1*2 - 1
      DO 30 I=1,N
      CALL DBLK(A1,B1,NA,NB,IB1,NFI1,IEOF1)
   30 CALL BLOCK(A1,BO,NA,NB,IBO,NFO)
      HRKEY2 = A1(2)
      CALL DBLK(A1,B1,NA,NB,IB1,NFI1,IEOF1)
      CALL BLOCK(A1,BO,NA,NB,IBO,NFO)
      HRINC = ABS(A1(2) - HRKEY2)
```

```
      IRINC = HRINC
C
C   COPY NAME RECORDS FROM NFI2 TO NFO, CHANGING NODE CODE AND KEY WORDS
C
      IRKEY1 = IA1(1)
      HRKEY2 = A1(2)
      DO 40 I=1,NTN2
      CALL DBLK(A2,B2,NA,NB,IB2,NFI2,IEOF2)
      IA2(1) = IRKEY1 + IRINC
      A2(2) = HRKEY2 - HRINC
      IRKEY1 = IA2(1)
      HRKEY2 = A2(2)
C  OFFSET NODE CODE
      A2(3) = A2(3) + OFFSET
      CALL BLOCK(A2,BO,NA,NB,IBO,NFO)
      CALL DBLK(A2,B2,NA,NB,IB2,NFI2,IEOF2)
      IA2(1) = IRKEY1 + IRINC
      A2(2) = HRKEY2 - HRINC
      IRKEY1 = IA2(1)
      HRKEY2 = A2(2)
      CALL BLOCK(A2,BO,NA,NB,IBO,NFO)
   40 CONTINUE
C
C   TRANSACTION NAME TABLE COMPLETED
C
C
C   TRANSFER TRANSACTION RECORDS FROM NFI1 TO NFO
C
   50 CALL DBLK(A1,B1,NA,NB,IB1,NFI1,IEOF1)
      IF (IEOF1 .GT. 1) GO TO 60
      NRI1 = NRI1 + 1
      CALL BLOCK(A1,BO,NA,NB,IBO,NFO)
      NRO = NRO + 1
      GO TO 50
C
C   COPY TRANSACTION RECORDS FROM NFI2 TO NFO, ADDING NTN1 TO TRANSACTION
C   INDICES SO THEY CORRESPOND TO THE NEW NAME TABLE ON NFO.
C
   60 CALL DBLK(A2,B2,NA,NB,IB2,NFI2,IEOF2)
      IF (IEOF2 .GT. 1) GO TO 70
      NRI2 = NRI2 + 1
      IA2(1) = IA2(1) + NTN1
      CALL BLOCK(A2,BO,NA,NB,IBO,NFO)
      NRO = NRO + 1
      GO TO 60
C
C   FILE MERGE COMPLETE
C
   70 CALL ENDB(BO,NB,NA,IBO,NFO)
      REWIND NFO
      WRITE(6,2000) NTN1,NTN2,NTNO,NRI1,NRI2,NRO,TBEG1,TBEG2,TBEG,
     *              TFIN1,TFIN2,TFIN
 2000 FORMAT(1H0,30X,6HFILE-1,6X,6HFILE-2,6X,6HOUTPUT  /
     *       25H NO. OF TRANSACTION TYPES,3I12   /
     *       25H NO. OF DATA RECORDS.....,3I12   /
     *       25H BEGINNING TIME.........,3F12.3 /
     *       25H ENDING TIME............,3F12.3 )
```

```
      WRITE(6,2010) OFFSET
 2010 FORMAT(//49H THE NODE CODES FOR TRANSACTIONS FROM FILE-2 ARE  ,
     *          16H INCREMENTED BY ,F10.3,20H ON THE OUTPUT FILE  )
      CALL EXIT
      END
```

C. CHTFH PROGRAM

1. Purpose

This utility program provides the capability to change transaction file header names given the index of the name. Thus the type, PERMAT resource ID, resource, or node number can be changed for any transaction on the file. This can be helpful to separate two transaction entries with different indices but the same name. This situation could occur due to a merging of two different model files or an error in model statistics collection.

2. Use

The program reads a transaction file in from file TAPE1 and writes out the transaction file on TAPE2 with only the specified names changed. The CHTFH program reads in a deck of input cards starting with the standard ODPS TITLE card used to verify the identity of the transaction file. This is followed by as many CHGNAM cards as are necessary to make the changes. These cards must be input in transaction index order. The final card is an end card. The formats of the CHTFH cards are shown in Table IV-2.

TABLE IV-2. FORMAT OF CHTFH CARDS

CARD COLUMN	FORMAT	DESCRIPTION
TITLE CARD - ONE CARD AT FRONT OF DECK		
1 - 5	A5	CARD IDENTIFICATION
6	1X	UNUSED
7	A1	CHECK OPTION[a]
8	1X	UNUSED
9 -10	I2	MONTH OF MODEL DATE
11	1X	UNUSED
12-13	I2	DAY OF MODEL DATE
14	1X	UNUSED
15-18	I4	YEAR OF MODEL DATE
19-20	2X	UNUSED
21-38	3A6	MODEL RUN NAME
39-40	2X	UNUSED
41-52	2A6	MODEL NAME
53-54	2X	UNUSED
55-66	2A6	ANALYST NAME
67-80	14X	UNUSED
CHGNAM CARDS - ONE FOR EACH INDEX NAME TO BE CHANGED		
1 - 6	A6	ENTER "CHGNAM"
11-20	I10	TRANSACTION INDEX
21-30	I10	TYPE CODE
31-40	I10	PERMAT RESOURCE
41-50	I10	RESOURCE NUMBER
51-60	I10	NODE NUMBER
END CARD - ONE CARD AT END OF DECK		
1 - 6	A6	ENTER "***END"

[a]IF THE CHECK OPTION FIELD IS NOT BLANK THEN NO COMPARISON IS MADE BETWEEN THE DATA ON THE TITLE CARD AND THE DATA ON THE INPUT FILE.

```
      PROGRAM CHTFH(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE1,TAPE2)
C
C     THIS PROGRAM CHANGES TRANSACTION FILE HEADER NAMES GIVEN THE INDEX
C     OF THE NAME.  THUS THE TYPE, PERMAT-RESID, RESOURCE, OR NODE NUMBER
C     CAN BE CHANGED FOR ANY TRANSACTION ON THE FILE. THIS CAN BE HELPFUL
C     TO SEPARATE TWO TRANSACTION ENTRIES WITH DIFFERENT INDEXES BUT THE
C     SAME NAMES.
C
C     FILES  TAPE1    INPUT FILE IN ASCENDING OR DESCENDING ORDER
C            TAPE2    OUTPUT FILE WITH ONLY THE SPECIFIED NAMES CHANGED
C
C     INPUT CARDS -
C
C          TITLE CARD AS SPECIFIED FOR AGATE OR TRAFAN
C
C          CHANGE CARDS - BLANK OR ZERO COMPONENTS WILL NOT BE CHANGED
C
C            CC1 - CC6    A6  CHGNAM -- CARD TITLE
C            CC11- CC20   I10 TRANSACTION INDEX
C            CC21- CC30   I10 TYPE
C            CC31- CC40   I10 PERM-RES
C            CC41- CC50   I10 RESOURCE
C            CC51- CC60   I10 NODE
C
C          END CARD
C
C            CC1 -CC6 A6 ***END
C
C          CARDS MUST BE ENTERED IN TRANSACTION INDEX ORDER
      COMMON /LINCOM/LINE,IPGSKP,ICLHR
      COMMON /GSPDUM/ TBEG,TFIN,NCLCT,MON,NDAY,NYR,RUNNAM(3),
     *                MODNAM(2),NAME(2),NMDPP(5)
      DIMENSION AI(4),IAI(4),BI(500),BO(500)
      EQUIVALENCE  (AI,IAI)
      DIMENSION TCTNAM(4,1500),CHGNAM(4,300), INDCHG(300)
      INTEGER TCTNAM,CHGNAM
      DIMENSION IMAP(4)
      DATA  NB,NA,NFI,NFO,MODE  /500,4,1,2,0 /
      DATA  END,ACHNAM    /6H***END,6HCHGNAM /
      DATA  LTCT,ISTNAM,IPTNAM,LCHG  /1500,1,1,300 /
      DATA  IMAP /3,2,4,1/
      DATA NMDPP /6HCHANGE,6H TRANS,6HACTION,6H FILE ,6HHEADER /
C
C  START TRANSACTION FILE READ, PLACING NAMES IN TCTNAM
C
      CALL IZRSTF(BI,NB,IB,NFI,IEOFI,ISTNAM,TCTNAM,LTCT,IPTNAM)
C
      CALL VERIFY
C
C  READ IN CHANGES, ASSUMED IN TRANS. INDEX ORDER
C
      ICHG = 0
   10 ICHG = ICHG + 1
      READ (5,1000) CRDNAM, INDCHG(ICHG),(CHGNAM(I,ICHG),I=1,4)
 1000 FORMAT(A6,4X,5I10)
C
      IF (CRDNAM .EQ. END) GO TO 30
```

```
      IF (CRDNAM .EQ. ACHNAM .AND. ICHG .LT.LCHG) GO TO 10
      WRITE (6,2000) CRDNAM, ICHG
 2000 FORMAT(18H WRONG CARD NAMED ,A6,27H OR TOO MANY CHGNAM CARDS.
     1        I5,12H CARDS READ.  )
   30 ICHG = ICHG - 1
C  ALL CHANGE CARDS READ, CHANGE SPECIFIED NAMES IN TCTNAM
      IF (ICHG .LE. 0) GO TO 100
      DO 50 I=1,ICHG
      J = INDCHG(I)
      DO 40 K=1,4
      IF (CHGNAM(K,I) .EQ. 0) GO TO 40
      KM = IMAP(K)
      TCTNAM(KM,J) = CHGNAM(K,I)
   40 CONTINUE
      WRITE (6,2010) J,TCTNAM(3,J),TCTNAM(2,J),TCTNAM(4,J),TCTNAM(1,J)
 2010 FORMAT(22H TRANSACTION NAME NO. ,I5,10HCHANGED...,5(I10,2X))
   50 CONTINUE
C
C  CHANGES COMPLETED IN TCTNAM, CREATE OUTPUT FILE WITH NEW NAME TABLE
C
      LINE = 60
      MODE = 2
      CALL INITB(BI,NB,IBI,NFI,MODE,IEOFI)
      MODE = 1
      CALL INITB(BO,NB,IBO,NFO,MODE,IEOFO)
C
C  COPY HEADER RECORDS
C
      DO 60 I=1,8
      CALL DBLK(AI,BI,NA,NB,IBI,NFI,IEOFI)
      IF (IEOFI .NE. 1) GO TO 200
      CALL BLOCK(AI,BO,NA,NB,IBO,NFO)
   60 CONTINUE
C
C  CHANGE NAME TABLE
C
      DO 70 I=1,LTCT
      CALL DBLK(AI,BI,NA,NB,IBI,NFI,IEOFI)
      IF (IEOFI .NE. 1) GO TO 200
      AI(3) = TCTNAM(1,I)
      AI(4) = TCTNAM(2,I)
      CALL BLOCK(AI,BO,NA,NB,IBO,NFO)
      CALL DBLK(AI,BI,NA,NB,IBI,NFI,IEOFI)
      IF (IEOFI .NE. 1) GO TO 200
      AI(3) = TCTNAM(3,I)
      AI(4) = TCTNAM(4,I)
      CALL BLOCK(AI,BO,NA,NB,IBO,NFO)
   70 CONTINUE
C
C  TRANSFER TRANSACTION RECORDS
C
   80 CALL DBLK(AI,BI,NA,NB,IBI,NFI,IEOFI)
      IF (IEOFI .NE. 1) GO TO 90
      CALL BLOCK(AI,BO,NA,NB,IBO,NFO)
      GO TO 80
C
C  FILE COMPLETE
```

```
C
   90 CALL ENDB(BO,NB,NA,IBO,NFO)
      REWIND NFO
      WRITE (6,2090)
 2090 FORMAT(48H TRANSACTION FILE HEADER NAME CHANGES COMPLETED.  )
C
      CALL IZRSTF(BO,NB,IB,NFO,IEOF,ISTNAM,TCTNAM,LTCT,IPTNAM)
  100 CALL EXIT
  200 WRITE (6,2200)
 2200 FORMAT(//45H END OF FILE HIT WHILE READING HEADER RECORDS    )
      CALL EXIT
      END
```

D. PROGRAM AGATE

The purpose of the Aggregation Program AGATE is to aggregate transactions across statistics type codes, resources, and nodes and produce an Aggregated Transaction File. The use of the program AGATE has not changed. Only certain aspects of the internal logic have been altered to handle continuous flow statistics, to handle statistics that are deactivated during a model run, and to set the sort keys on the transaction name table.

The program was altered to change the assumption that a change in a TMST statistic occurred for every entry on the transaction file. This was necessary for continuous flow statistics that are collected on a periodic basis, whether the variable changes or not.

When statistics variables are deactivated, the AGATE program must realize this and not spread the latest value over the remaining time. To recognize this, an expected time for the next transaction for an aggregated component is stored so that an inactive statistic may be terminated.

The transaction file header name table has two sort keys at the start of every record. The first key is an integer value which causes the name table to be placed at the beginning of the file by an ascending sort on the transaction index for each record. The second key is now a real value which causes the name table entries to be placed at the beginning by a descending sort on the transaction record time values.

```
      PROGRAM AGATE (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,
     *               TAPE1,TAPE2,TAPE3)
C
C     OUTPUT DATA PROCESSOR AGGREGATION PROGRAM
C
C     PURPOSE----
C           THIS COPIES A TRANACTION FILE ONTO AN AGGREGATED
C     TRANSACTION FILE.  THE TRANSACTIONS OF A GIVEN TYPE FOR
C     MULTIPLE NODES AND RESOURCE MAYBE AGGREGATED INTO ONE
C     DEFINED BY THE SAME TRANSACTION TYPE CODE AND USER DEFINED
C     AGGREGATION NODE AND RESOURCE.
C
C     INPUT
C       1. TRANSACTION FILE -- TAPE 1
C          (SORTED INTO DESCENDING TIME ORDER WITH THE FILE HEADER DATA
C           AND TRANSACTION DEFINITIONS AT THE BEGINNING.)
C       2. AGGREGATION DEFINITIONS -- CARDS(TAPE5)
C
C     OUTPUT
C       1. AGGREGATED TRANSACTION FILE --TAPE2
C       2. REPORT OF AGGREGATION TRANSACTION DEFINITIONS AND LARGEST
C          SIMULATION TIME VALUES FOR EACH COMPONENT.
C
C     INTERMEDIATE -- COMPONENT AND AGGREGATION TRANSACTION DEFINITION
C                     BY INDEX -- TAPE3
      DIMENSION BI(512), AI(4), BO(512), AO(4), IAI(4), IAO(4),
     *          INX(1500),NOTS(1500),XOC(1500),TOC(1500),TEC(1500),
     *          IXC(2000),IXA(2000),IXCA(2000),
     *          TOA(1000),XOA(1000),YOA(1000),INXC(1000),
     *          TCTNAM(4,1500), IAA(5), ICARD(8),
     *          NOCN(200), NOCR(200), NOCV(200),
     *          IDT(1500), IDIX(1500), NODPP(5)
      COMMON /WR/ NPRNT
      DATA NPRNT /6/
      COMMON /GSPDUM/ TBEG,TFIN,NCLCT,MON,NDAY,NYR,RUNNAM(3),
     *                MODNAM(2),NAME(2),NMDPP(5)
      EQUIVALENCE (AI,IAI), (AO,IAO), (AI,IXT),  (AI(2),T), (AI(3),X),
     *            (INX,TCTNAM), (XOC,TCTNAM(1, 376)),
     *            (XOA,TCTNAM(1, 751)), (TOA,TCTNAM(1,1126)),
     *            (AI(4),Y)
      DATA IAA/3HAGG,3HNOD,3HRES,3HEND,3HVAR/
      DATA NODPP/1H ,6HAGGREG,6HATION ,6HPROGRA,1HM/
      INTEGER TCTNAM
C     CONSTANTS
      LTCT = 1500
      NA = 4
       NB = 500
      IPTNAM = 1
      ISTNAM = 1
      LATCT = 1000
      LCART = 2000
      NOFI = 1
      NOFO = 2
      DO 1 I=1,5
```

```
1       NMOPP(I) = NOOPP(I)
C       START TRANS ACTION FILE
        CALL IZRSTF(BI,NB,IB,NOFI,IEOFI,ISTNAM,TCTNAM,LTCT,IPTNAM)
        CALL VERIFY
C       WRITE COMPONENT TRANSACTION DEFINITIONS ONTO SCRATCH FILE
        REWIND 3
        WRITE (3) ((TCTNAM(I,J),I=1,4),J=1,LTCT)
C
C
C
        DO 200 I=1,LTCT
        NOTS(I) = MOD(TCTNAM(3,I)/10,10)
        IF (NOTS(I) .NE. 2 .AND. NOTS(I) .NE. 3) NOTS(I) = 1
        IDT(I) = TCTNAM(3,I)
200     IDIX(I) = I
C
C       SORT IDT.
        NP = LTCT
        L  = LTCT
202     L  = (L+1)/2
        DO 206 II=1,NP
        I = II
        J = II+L
        IF (J .GT. NP) GO TO 208
204     IF (IDT(I) .LE. IDT(J)) GO TO 206
        IH = IDT(I)
        IDT(I) = IDT(J)
        IDT(J) = IH
        IH = IDIX(I)
        IDIX(I) = IDIX(J)
        IDIX(J) = IH
        J = I
        I = I-L
        IF (I .GT. 0) GO TO 204
206     CONTINUE
208     IF (L .GT. 1) GO TO 202
C
C       STORE AGGREGATION DEFINITIONS
        NOA = 0
        NIL = 0
        NOBC = 0
        IIN = 0
218     READ (5,220) ICARD
219     FORMAT (2X,A3,7X,7I10)
        PRINT 219, ICARD
220     FORMAT (A3,7X,7I10)
        DO 222 I=1,5
        IF (ICARD(1) .EQ. IAA(I)) GO TO 226
222     CONTINUE
C       BAD CARD.
223     WRITE (6,224) ICARD
224     FORMAT (//17H IMPROPER CARD =,A3,7X,7I10,17H= OR OUT OF ORDER)
        NOBC = NOBC+1
        GO TO 218
226     GO TO (230,260,270,280,274),I
C       AGGREGATION CARD.
230     ATYPE = 1
```

```
      WRITE (3) ICARD(2), ICARD(3), ICARD(4)
231   IF (IIN .EQ. 0 .OR. IIR .EQ. 0) GO TO 252
      IF (IIV .NE. 0) GO TO 227
      NOCV(1) = NATT
      IIV = 1
227   IF (IIV .EQ. 1) GO TO 229
      I = MOD(NOCV(1)/10,10)
      DO 228 J=2,IIV
      IF (I .NE. MOD(NOCV(J)/10,10)) GO TO 252
228   CONTINUE
229   DO 253 K=1,IIV
      NATT = NOCV(K)
C     FIND FIRST COMPONENT WITH AGGREGATE TYPE TRANSACTION.
      N1 = 1
      N2 = LTOT
232   IX = (N1+N2)/2
234   IF (NATT - IDT(IX)) 236,244,238
C     NATT IS SMALLER
236   IF (IX .EQ. N1) GO TO 240
      N2 = IX
      GO TO 232
C     NATT IS LARGER
238   IF (IX .EQ. N2) GO TO 240
      N1 = IX
      IX = (N1+N2+1)/2
      IF (N1+1 .EQ. N2) GO TO 239
      GO TO 234
239   IX = N1
      IF (NATT .EQ. IDT(N1)) GO TO 244
      IX = N2
      IF (NATT .EQ. IDT(N2)) GO TO 244
C     NO MATCH.
240   WRITE (6,242) NATT
242   FORMAT (//19H TRANSACTION TYPE *,I10,13H* NOT ON FILE)
      GO TO 253
244   IF (IX .EQ. 1) GO TO 246
      IF (IDT(IX) .NE. IDT(IX-1)) GO TO 246
      IX = IX-1
      GO TO 244
C
246   IC = IDIX(IX)
      DO 250 I=1,IIN
      IF (NOCN(I) .NE. TOTNAM(1,IC)) GO TO 250
      DO 248 J=1,IIR
      IF (NOCR(J) .NE. TOTNAM(4,IC)) GO TO 248
      NIL = NIL+1
      IXC(NIL) = IC
      IXA(NIL) = NOA
      GO TO 251
248   CONTINUE
250   CONTINUE
251   IX = IX+1
      IF (IDT(IX) .EQ. IDT(IX-1)) GO TO 246
253   CONTINUE
      IF (ICARD(1) .EQ. -1) GO TO 282
C
252   NATT = ICARD(2)
```

```
      NOA = NOA+1
      IIV = 0
      IIN = 0
      IIR = 0
      IF (ICARD(1) .EQ. -1) GO TO 282
      GO TO 218
C     NODE CARD.
260   DO 262 I=2,8
      IF (ICARD(I) .EQ. 0) GO TO 262
      IIN = IIN+1
      NOCN(IIN) = ICARD(I)
262   CONTINUE
      GO TO 218
C     RESOURCE CARD
270   DO  272 I=2,8
      IF (ICARD(I) .EQ. 0) GO TO 272
      IIR = IIR+1
      NOCR(IIR) = ICARD(I)
272   CONTINUE
      GO TO 218
C     STATISTICS TYPE CODE CARD.
274   DO 275 I=2,8
      IF (ICARD(I) .EQ. 0) GO TO 275
      IIV = IIV+1
      NOCV(IIV) = ICARD(I)
275   CONTINUE
      GO TO 218
C
C
280   ICARD(1) = -1
      GO TO 231
C
C
282   END FILE 3
      REWIND 3
      IF (NOBC .NE. 0) STOP
C
C  SAVE RELATION TABLE IN AGG SORT ORDER AND SET UP POINTERS TO NEXT
C  COMPONENTS FOR EACH AGGREGATE  ( INXC ARRAY )
      DO 283 I=1,NIL
283   IXCA(I) = IXC(I)
      IX = 1
      DO 288 I=1,NOA
284   INXC(I) = IX
      IF (IX .GT. NIL) GO TO 288
      IF (IXA(IX) .EQ. I) GO TO 286
      IF (I .LT. IXA(IX)) GO TO 288
      IX = IX + 1
      IF (IX .GT. NIL) GO TO 288
      GO TO 284
286   IX = IX + 1
      IF (IX .GT. NIL) GO TO 288
      IF (IXA(IX) .EQ. I ) GO TO 286
288   CONTINUE
      INXC(NOA + 1) = IX
C  INXC NOW POINTS TO START OF COMPONENTS IN AN AGGREGATE IN LIST
C  CONTAINED IN IXCA ARRAY
```

```
C       SORT RELATION TABLE.
        NP = NIL
        L  = NIL
290     L  = (L+1)/2
        DO 294 II=1,NP
        I = II
        J = II+L
        IF (J .GT. NP) GO TO 296
292     IF (IXC(I) .LE. IXC(J)) GO TO 294
        IH = IXC(I)
        IXC(I) = IXC(J)
        IXC(J) = IH
        IH = IXA(I)
        IXA(I) = IXA(J)
        IXA(J) = IH
        J = I
        I = I-L
        IF (I .GT. 0) GO TO 292
294     CONTINUE
296     IF (L .GT. 1) GO TO 290
C
        TENDP = TFIN+100.0
        IX = 1
        DO 302 I=1,LTOT
        TOC(I) = TENDP
        TEC(I) = 0.0
        XOC(I) = 0.0
297     INX(I) = IX
        IF (IX .GT. NIL) GO TO 298
        IF (IXC(IX) .EQ. I) GO TO 300
        IF (I .LT. IXC(IX)) GO TO 298
        IX = IX+1
        IF (IX .GT. NIL) GO TO 298
        GO TO 297
298     NOTS(I) = 4
        GO TO 302
300     IX = IX+1
        IF (IX .GT. NIL) GO TO 302
        IF (IXC(IX) .EQ. I) GO TO 300
302     CONTINUE
        INX(LTOT+1) = IX
C
C
        DO 304 I=1,NOA
        TOA(I) = TENDP
        YOA(I) = 0.0
304     XOA(I) = 0.0
C
C       START OUTPUT FILE.
        MODE = 1
        CALL INITB(BO,NB,IBO,NOFO,MODE,IEOFO)
        IAO(1) = -1.0E+10
        AO(2) = 1.0E+10
        DO 324 I=1,7
        IAO(1) = IAO(1)+1
        AO(2) = AO(2) - 1.
        GO TO (310,312,322,314,316,318,320), I
```

```
310     IAD(3) = 6HMAWLOG
        IAD(4) = 6HS AGAT
        GO TO 324
312     IAD(3) = MODNAM(1)
        IAD(4) = MODNAM(2)
        GO TO 324
314     IAD(3) = MON
        IAD(4) = NDAY
        GO TO 324
316     IAD(3) = NYR
        IAD(4) = RUNNAM(1)
        GO TO 324
318     IAD(3) = RUNNAM(2)
        IAD(4) = RUNNAM(3)
        GO TO 324
320     IAD(3) = TBEG
        IAD(4) = TFIN
        GO TO 324
322     IAD(3) = NAME(1)
        IAD(4) = NAME(2)
324     CALL BLOCK (AD,BD,NA,NB,IBD,NOFD)
C
        CALL PGHDRO
C                                  ***
C       EXTRACT AND AGGREGATE      ***
C                                  ***
C
400     CALL DBLK (AI,BI,NA,NB,IB,NOFI,IEOFI)
        IF (IEOFI.NE.1) GO TO 500
        IF (NOTS(IXT) .EQ. 4) GO TO 400
        IXIN = NOTS(IXT)
        IXB = INX(IXT)
        IXE = INX(IXT+1)-1
        IF (TOC(IXT) .EQ. TENDP) TOC(IXT) = T
        GO TO (410,450,450), IXIN
C       TYPE STATISTIC = 1.
410     DO 425 I=IXB,IXE,1
        IX = IXA(I)
        IF (TOA(IX) .LE. TFIN) GO TO 415
        TOA(IX) = T
        XOA(IX) = X
        YOA(IX) = Y
        GO TO 425
C
415     IF (TOA(IX) .EQ. T) GO TO 420
C
C   TIME CHANGE, SUBTRACT ANY COMPONENT X VALUES THAT HAVE NO
C   TRANSACTIONS BEFORE THIS TIME
        XOAIX = XOA(IX)
        IXXB = INXC(IX)
        IXXE = INXC(IX + 1) - 1
        DO 417 II = IXXB,IXXE,1
        IXX = IXCA(II)
        IF(T .GE. TEC(IXX)) GO TO 417
C   COMPONENT IXX HAS NO EARLIER ENTRIES
        XOA(IX) = XOA(IX) - XOC(IXX)
  417 CONTINUE
```

```
      IAO(1) = Ix
      AO(2) = TOA(IX)
      AO(3) = xOA(IX)
      AO(4) = TOA(IX)-T
      CALL BLOCK (AO,BO,NA,NB,IBO,NOFO)
C
      TOA(IX) = T
C  RESTORE XOA(IX) TO FULL VALUE INCLUDING ANY INACTIVE COMPONENT VALUES
      XOA(IX) = XOAIx
      YOA(Ix) = Y
420   XOA(Ix) = XOA(IX)+(X-XOC(IXT))
425   CONTINUE
C
      XOC(IXT) = X
C  SET EXPECTED TIME FOR NEXT TRANSACTION FOR THIS COMPONENT
      TEC(IXT) = T + Y
      GO TO 400
C     TYPE STATISTIC 2 AND 3.
450   DO 460 I=IXB,IxE,1
      IAO(1) = IxA(I)
      AO(2) = T
      AO(3) = x
      AO(4) = Y
460   CALL BLOCK (AO,BO,NA,NB,IBO,NOFO)
      GO TO 400
C     EXTRACT COMPLETE
500   DO 510 I=1,NOA
      IF (TOA(I) .EQ. TENOP) GO TO 510
C  SUBTRACT ANY COMPONENT X VALUES THAT ARE INACTIVE
      IxxB = INXC(I)
      IXXE = INXC(I + 1) - 1
      DO 505 II = IxxB,IXXE,1
      Ixx = IXCA(II)
      IF (TOA(I) .GE. TEC(IXX)) GO TO 505
C  COMPONENT IXX IS INACTIVE
      XOA(I) = XOA(I) - XOC(IXX)
  505 CONTINUE
      IAO(1) = I
      AO(2) = TOA(I)
      AO(3) = XOA(I)
      AO(4) = YOA(I)
      CALL BLOCK (AO,BO,NA,NB,IBO,NOFO)
510   CONTINUE
C
C
      READ (3) ((TCTNAM(I,J),I=1,4), J=1,LTCT)
C
      IAO(1) = -1.0E+10 + 8
      AO(2) = 1.0E+10-8.
      IAO(3) = NOA
      IAO(4) = 0
      CALL BLOCK (AO,BO,NA,NB,IBO,NOFO)
      L = 61
      Ix = 1
      DO 622 I=1,NOA
      READ (3) NATT, NATN, NATR
      IAO(1) = IAO(1)+1
```

```
      AO(2) = AO(2) - 1.
       AO(3) = NATN
       AO(4) = 0
      CALL BLOCK (AO,BO,NA,NB,IBO,NOFO)
      IAO(1) = IAO(1)+1
      AO(2) = AO(2) - 1.
       AO(3) = NATT
       AO(4) = NATR
      CALL BLOCK (AO,BO,NA,NB,IBO,NOFO)
C
C  LIST COMPONENTS OF THIS AGGREGATE
      IXB = INXC(I)
      IXE = INXC( I + 1) - 1
C
      IHDR = 0
      DO 620 J=IXB,IXE,1
      IC = IXCA(J)
      IT = TOC(IC)
      IF (IT.GT.TFIN) GO TO 620
      IF (L.LT.61) GO TO 615
608   CALL PGHDRO
      IHDR = 0
      WRITE (6,610)
610   FORMAT (1H0,5X,23HAGGREGATION TRANSACTION,25X,10HCOMPONENT ,
     *        12HTRANSACTIONS//6X,18HINDEX   TRANS-TYPE,8X,4HNODE,
     *        4X,8HRESOURCE,6X,18HINDEX   TRANS-TYPE,8X,4HNODE,4X,
     *        8HRESOURCE,2X,10HTIME(LAST))
612   FORMAT (1H0,6X,I4,1X,3I12,7X,I4,1X,4I12)
614   FORMAT (54X,I5,1X,4I12)
      L = 9
      GO TO 616
  615 IF (IHDR .EQ. 1) GO TO 618
      IF (L .GT. 58) GO TO 608
616   WRITE (6,612) I,NATT,NATN,NATR,IC,TCTNAM(3,IC),TCTNAM(1,IC),
     *              TCTNAM(4,IC),IT
      IHDR = 1
      L = L+2
      GO TO 620
618   WRITE (6,614)IC,TCTNAM(3,IC),TCTNAM(1,IC),TCTNAM(4,IC),IT
      L = L+1
620   CONTINUE
622   CONTINUE
      CALL PGHDRO
      CALL ENDB(BO,NB,NA,IBO,NOFO)
C
      REWIND NOFI
      REWIND NOFO
C
      STOP
      END
```

E. PROGRAM TRAFAN

The program TRAFAN is the controlling program for the Transaction File Analyzer Program. The basic purpose of this routine is threefold. First, it reads the input data cards and stores the information in the AGENDA and the GRPHDA labeled COMMON blocks. Second, TRAFAN reads the Transaction File and performs an extract for the variables specified in the card deck. Finally, it passes control to the appropriate subroutines to accomplish the statistical analysis and report generation.

The use of the TRAFAN program has not been changed. The only logic change was to properly spread the values of a TMST statistics over time periods for continuous flow model statistics.

```
      PROGRAM TRAFAN (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE1,TAPE2
     *                TAPE3,TAPE8)
C     TRANSACTION FILE ANALYZER
C     PURPOSE - TO GENERATE HISTOGRAMS, MEANS, STANDARD DEVIATIONS, MAXS
C               AND MINS ON SELECTED DATA
C
      DIMENSION IA(9),IC(6),A(4),B(500),ISEQ(16),JSEQ(16),SELCT(1500),
     *          IEXTX(5,200), IAA(12), LAB(1200,6),XCOUNT(2),
     *          COSTF(2), NODPP(5)
C
      COMMON /WR/ NPRNT
      DATA NPRNT /6/
      COMMON /AGENDA/ IX(1000), ISELCT(3000), NOSVS, IXMAX, ISMAX
      DATA IXMAX, ISMAX /1000,3000/
      COMMON /SEQI/ISV,ISN,MAXN,IGON,ISR,MAXR,IGOR,IT,ISP,ICODE,IV
     *             ,IN,IR,LENN,LENR,LENT
      COMMON /SEQJ/JSV,JSN,JMAXN,JGON,JSR,JMAXR,JGOR,JT,JSP,JCODE,JV
     *             ,JN,JR,JLENN,JLENR,JLENT
      COMMON /SVCNT/ SUMXCT(1000),NCOUNT,NXTCN
      DATA NCOUNT / 1000/
      COMMON /SVCLCT/ SUMXWT(1200),SUMX(1200),SUMXX(1200),XMIN(1200),
     *                XMAX(1200),NCOLCT,NXTCL
      DATA NCOLCT / 1200/
      COMMON /SVHSTO/ XLO, W, H(22, 320), NH( 320), NHMAX,
     *                NHISTO,NXTHST,F(22),CF(22),AVG,SD,MI
      DATA NHISTO, NHMAX / 320, 22/
      COMMON /GSPDUM/ TBEG,TFIN,NCLCT,MON,NDAY,NYR,RUNNAM(3),
     *                MODNAM(2),NAME(2),NMDPP(5)
      COMMON /GRPHDA/ IGRAPH(6,400), GB(300), LIG, LGB, LGA,
     *                LINDXG, IGB, NGRPHF, IEOFG, GA(3)
      DATA LIG, LGB, LGA, NGRPHF / 400,300,3,3/
      COMMON /INSTAT/IXX,X1,X2,X3
C
      INTEGER TCTNAM(4,1500), GA,RUNNAM
C
      EQUIVALENCE (ISEQ(1),ISV),
     *            (JSEQ(1), JSV),
     *            (ISELCT,SELCT),
     *            (A(1),ITXTF),  (A(2),TNOW),
     *            (TCTNAM,SUMXWT),
     *            (LAB,XLO), (MAXL1,AVG),(MAXL2,SD),(MAXL3,MI),
     *            (X1,XCOLCT,XCOUNT   ,XHISTO),
     *            (X2,XTCLCT,          XTHSTO), (IA,IAA)
C
      DATA NFILE, MODE/ 1, 1/,
     *     IHIS/6HHISTO /,IC/          6HTIME  ,6HRES   ,6HNODES ,
     *     3HVAR,3HEND,5HGRAMG/, IGNAM/5HGRAPH/,IBLANK/1H /,
     *     MAXTYP,ISING,ICOMB,IALL/6,6HSINGLE,4HCOMB,3HALL/,
     *     IPRIM,ISEC /6HPRIM  ,6HSECOND /, IGRAM /6HGRAM  /
C
      DATA LTCT,NA,NB,ISTNAM,IPTNAM/
     *     1500, 4, 500, 1, 1/
      DATA COSTF, NODPP/6HCOST  ,6HFILE  ,6H   TRA,6HNSACTI,6HON FIL,
     *                  6HE ANAL,6HYZER  /, NONS/6HNONSEQ/
      DO 5 I=1,5
5     NMDPP(I) = NODPP(I)
C     INITIALIZE TRANSACTION FILE
```

```
      CALL IZRSTF (B,NB,IB,NFILE,IEOF,TSTNAM,TCTNAM,LTCT,IPTNAM)
      CALL VERIFY
C     INIALIZE GRAPH FILE.
      CALL INITB (GB,LGB,IGB,NGRPHF,MODE,IEOFG)
C     STORE GRAPH TITLE DATA.
  200 READ (5,202) IAA
  202 FORMAT (A6,4X,2I5,1X,A6,1X,8A6)
      IF (IA(1) .NE. IGNAM) GO TO 210
      GA(1) = IAA(2)*1000+IAA(3)*10
      IF (IAA(3) .EQ. 0 .AND. IAA(4) .EQ. IBLANK) GO TO 206
      GA(1) = GA(1)+100
      IF (IAA(3) .NE. 0) GA(1)=GA(1)+100
      DO 204 I=1,3,2
      GA(1) = GA(1)+1
      GA(2) = IAA(I+3)
      GA(3) = IAA(I+4)
  204 CALL BLOCK (GA,GB,LGA,LGB,IGB,NGRPHF)
      GO TO 200
C
  206 DO 208 I=1,7,2
      GA(1) = GA(1)+1
      GA(2) = IAA(I+4)
      GA(3) = IAA(I+5)
  208 CALL BLOCK (GA,GB,LGA,LGB,IGB,NGRPHF)
      GO TO 200
C
  210 IXIG = 0
      GA(1) = 0
      DO 222 I=1,8
      GA(1) = GA(1)+1
      GO TO (212,214,216,218,220,217,219,221), I
  212 GA(2) = MON
      GA(3) = NDAY
      GO TO 222
  214 GA(2) = NYR
      GA(3) = RUNNAM(1)
      GO TO 222
  216 GA(2) = RUNNAM(2)
      GA(3) = RUNNAM(3)
      GO TO 222
  218 GA(2) = MODNAM(1)
      GA(3) = MODNAM(2)
      GO TO 222
  220 GA(2) = NAME(1)
      GA(3) = NAME(2)
      GO TO 222
  217 GA(2) = NODPP(1)
      GA(3) = NODPP(2)
      GO TO 222
  219 GA(2) = NODPP(3)
      GA(3) = NODPP(4)
      GO TO 222
  221 GA(2) = NODPP(5)
  222 CALL BLOCK(GA,GB,LGA,LGB,IGB,NGRPHF)
C     WRITE HEADER RECORD OF COST FILE.
      REWIND 8
      II = 0
```

```
      IS = 12
      WRITE (8) II,II,IS,COSTF,MODNAM,NAME,MON,NDAY,NYR,RUNNAM
C     HAVE HISTO CARD AND NHMAX
      IF(IA(1).NE.IHIS) CALL ERROR(1,IA)
C     INITIALIZE
      NOSCTF = 0
      NOSVS = 0
      IXSEQ=0
      II=1
      IS=1
      KTYPE=0
      NHMAX = IA(3)
C     READ GENERAL TIME CARD.
      READ (5,591) IA
      IF (IA(1) .NE. IC(1)) STOP
      ITANF = IA(3)
      ITHALT = IA(4)
      ITLEN = IA(5)
      DO 223 I=1,LIG
      DO 223 J=1,6
  223 IGRAPH(J,I)=0
C     II IS THE INDEX IN IX(IS)
C     IS IS THE INDEX IN SELECT(IS)
C     KTYPE INDICATES THE SELECTOR TYPE
C     1, IF A VARIABLE CARD HAS BEEN READ
C     2,       NODE
C     3,       RESOURCE
C     4,       TIME PERIOD
C     KOUNT  IS THE NUMBER OF VALUES OF A CERTAIN CATEGORY OF SELECTION
C            WHICH APPEAR ON A DATA CARD
C     IX(II) IS THE ARRAY WHICH CONTAINS SELECTOR COUNTS AND POINTERS
C            TO THE ARRAY SELECT
C     ISELCT(IS) IS THE ARRAY WHICH CONTAINS HISTOGRAM SELECTOR VALUES
      IVX=3
      INX=0
      IRX=0
      LABOLD=1
      LABNEW=1
C     READ TIME, RES,NODES, VAR OR END CARD
   10 READ(5,591)IA
  591 FORMAT (2A6,I8,6I10)
C     CHECK FOR DATA CARD TYPE
      DO 20 J=1,MAXTYP
      JTYPE=J
      IF(IA(1) .EQ. IC(J)) GO TO 30
   20 CONTINUE
      IF (IA(1) .NE. NONS) GO TO 25
      NOSCTF = 1
      GO TO 10
25    CALL ERROR(2,IA(1))
      GO TO 10
   30 GO TO (600,500,400,300,1000,630), JTYPE
C     VAR (VARIABLE) CARD
  300 IF(KTYPE .NE. 0)GO TO 350
      KTYPE=1
  305 KOUNT=0
  308 IX(II)=IS
```

```
      IF (NOSCTF .EQ. 0 .AND. JTYPE .EQ. 4) NOSVS=NOSVS+1
      II=II+1
      IA2OLD=IA(2)
C     CHECK ENTRIES ON CARD
  310 DO 325 I=3,9
      IF(IA(I).LE. 0) GO TO 325
      KOUNT=KOUNT+1
      ISELCT(IS)= IA(I)
      IS=IS+1
  325 CONTINUE
      IF(IA(2).EQ. IPRIM) ISELCT(IS-1)=ISELCT(IS-1)/10000
      IF(KOUNT .GT. 0) GO TO 340
      IF(IA(1).EQ.IC(4).OR.IA(2).NE.IALL) CALL ERROR(3,IA(1))
      ISELCT(IS)=0
      IS=IS+1
      IX(II)=1
      GO TO 345
  340 IX(II)=KOUNT
  345 II=II+2
C     PRIM (WITH ALL SECONDARIES) AND SECOND (WITH ALL PRIMARIES)
C       MAY NOT HAVE MORE THAN ONE ELEMENT SPECIFIED.
      IF((IA(2).EQ.IPRIM .OR. IA(2).EQ.ISEC) .AND. KOUNT .GT. 1)
     * CALL ERROR(11, IA(1))
      IF(KOUNT.LE.0) GO TO 10
      IF(IA(2) .EQ. IALL) CALL ERROR(13,IA(1))
      GO TO 10
C     IS THIS A VAR CONTINUATION CARD
  350 IF (KTYPE .NE. 1) GO TO 370
      II=II-2
      GO TO 310
  370 IF( KTYPE .NE. 4)CALL ERROR(4,KTYPE)
C     NEW VAR SET, SET REPEAT CODE
      IX(II)=1
      II=II+1
      KTYPE=1
      IX(IVX)= II
      IVX=II+2
      GO TO 305
C     NODES CARD
  400 IF( KTYPE .NE. 1)GO TO 450
C     NEW NODES SET
  405 IF (INX.NE.0) IX(INX) =999
  406 INX=II+3
      KTYPE=2
C     CHECK FOR SINGLE, COMB, OR ALL
  407 IF(IA(2).NE.ISING)GO TO 420
      IX(II)=1
      II=II+1
      GO TO 305
  420 IF(IA(2).NE.ICOMB)GO TO 430
      IX(II)=2
      II=II+1
      GO TO 305
  430 IF(IA(2).NE. IALL) CALL ERROR(5,IA(1))
      IX(II)=3
      II=II+1
      GO TO 305
```

```
C       NODE CONTINUATION
  450 IF(KTYPE .NE. 2) GO TO 470
  455   II=II-2
        IF(IA(2) .NE. IA2OLD)CALL ERROR(6,IA(1))
        IF(IA(2) .EQ. IALL ) CALL ERROR(7,IA(1))
        GO TO 310
C       NEW NODES CARD
  470 IF(KTYPE .NE. 4)CALL ERROR(4,KTYPE)
C       SET REPEAT CODE
        IX(II)=2
        II=II+1
        IX(INX)=II
        GO TO 406
C       RES (RESOURCE) CARD
  500 IF(KTYPE .NE. 2) GO TO 550
  505 IF(IRX .NE. 0)IX(IRX)=999
  506   IRX=II+3
        KTYPE=3
C       CHECK FOR PRIMARIES OR SECONDARIES
        IF(IA(2) .NE. IPRIM) GO TO 520
        IX(II) = 4
        II=II+1
        GO TO 305
  520 IF(IA(2) .NE. ISEC)GO TO 407
        IX(II)=5
        II=II+1
        GO TO 305
  550 IF(KTYPE .NE. 3) GO TO 570
C       RES CONTINUATION
        KA=IX(II-4)
C       PRIM (SECOND) CARD WITH ALL SECONDARIES (PRIMARIES) MAY NOT BE
C       CONTINUED
        IF(KA.EQ.4.OR.KA.EQ.5)  CALL ERROR(7,IA(1))
C       GO TO 455 TO MAKE MORE CHECKS ON CONTINUATIONS
        GO TO 455
  570 IF(KTYPE .NE. 4) CALL ERROR(4,KTYPE)
C       NEW RESOURCE CARD
C       SET REPEAT CODE
        IX(II)=3
        II=II+1
        IX(IRX)=II
        GO TO 506
C
C       TIME PERIOD CARD
  600 IF(KTYPE .NE. 3) CALL ERROR(8,IA(1))
        KTYPE =4
        IX(II)=IS
        II=II+1
        IF (ISIGN(1,IA(3)).LT.0) IA(3)=ITAVE
        IF (IA(4).LE.0) IA(4)=ITHALT
        IF(IA(5).LE.0) IA(5)=ITLEN
C       STARTING TIME OF FIRST TIME PERIOD.
        SELCT(IS) = IA(3)
        IS=IS+1
C       TIME INCREMENT I.E. LENGTH OF TIME PERIODS.
        SELCT(IS)=IA(5)
        IS=IS+1
```

```
C       NUMBER OF PERIODS.
        ISELCT (IS) = (IA(4) - IA(3))/IA(5)
C       SAFETY CHECK.
        IF (NOSCTF .EQ. 0) GO TO 605
        IF (ISELCT(IS) .LT. 1 .OR. ISELCT(IS) .GT. 99) ISELCT(IS)=1
605     IS=IS+1
C       NOW READ A GRAM CARD FOR HISTOGRAM DATA
        READ(5,592) IA(1),IA(2),XL ,WH,(IA(I),I=3,9)
  592   FORMAT ( A6,I4,2F10.5, 7A6)
        ICODE= IA(2)
C       STORE IN ISELCT
        ISELCT(IS)=ICODE
        IS=IS + 1
        IF(ICODE .LT.0 .OR. ICODE .GT. 4)CALL ERROR(12,ICODE)
        IF(IA(1)  .NE. IGRAM) CALL ERROR(9,IA(1))
        SELCT(IS) = XL
        IS=IS+1
        SELCT(IS) = WH
        IS=IS+1
        IMODE = 0
C       STORE ALPHA DESCRIPTOR DATA
        DO 620 I=3,9
        ISELCT(IS) = IA(I)
        IS=IS+1
  620   CONTINUE
        ISELCT(IS) = 0
        ISELCT(IS+1) = 0
        IS = IS+2
        GO TO 10
C       STORE GRAPH INDEX DATA
  630   IF (IXIG .NE. 0 .AND. ISELCT(IS-2) .EQ. 0) ISELCT(IENDGX) = IXIG
        IXIG = IXIG+1
        IENDGX = IS-1
        IF (ISELCT(IS-2) .EQ. 0) ISELCT(IS-2) = IXIG
        IGRAPH(1,IXIG) = IA(3)*1000+IA(4)*10+301
        DO 632 I=2,6
  632   IGRAPH(I,IXIG) = IA(I+3)
        IF (IA(5) .LT. 1 .OR. IA(5) .GT. 9) IGRAPH(2,IXIG)=1
        GO TO 10
C
C       END CARD
 1000   IX(II)=4
        IX(IVX)=999
        IX(INX)=999
        IX(IRX)=999
        IF (IXIG .NE. 0) ISELCT(IENDGX) = IXIG
        IF (IXIG .LE. LIG .AND. IS .LE. ISMAX .AND. II .LE. IXMAX)
       *GO TO 642
  638   WRITE (5,640) IS, ISMAX, II, IXMAX, IXIG, LIG
        STOP
  640   FORMAT (21H1SOME ARRAYS EXCEEDED/
       *        21H ISELCT--REQUIRED....,I4,16H  MAX ALLOWED....I4/
       *        21H IX-REQUIRED.........,I4,16H  MAX ALLOWED...,I4/
       *        21H IGRAPH--REQUIRED....,I4,16H  MAX ALLOWED...,I4)
  642   CONTINUE
C       LAB(I,J),I=INDEX, J=1 FOR COUNT, 2 FOR COLCT, 3 FOR HISTO LABELS
C       DETERMINE ALL POSSIBLE LABELS STORE IN LAB AND SORT.
```

```
      CALL LABALL (1)
C
      CALL WEXT(LTCT)
      IV = 1
      IGON = 1
      IN = 1
      IGOR = 1
      IR = 1
      ISV = 0
      LITX = 0
      ITXTF = 0
      GO TO 1060


C     READ TRANSACTION FILE
C  IT, JT, AND KT ARE TIME PERIOD INDEXES
C     IT = INDEX TO TP CURRENTLY BEING ACCUMULATED
C     JT = INDEX TO TP WHICH CURRENT RECORD SHOULD FALL IN
C     KT = INDEX TO TP TRANSACTION WAS PLACED ON TAPE IN
C  FOR TMST TYPE STATS (TYPSAT.EQ.1), THE ELAPSED TIME, A(4), IS
C  SUBTRACTED FROM TNOW TO DETERMINE WHAT TIME PERIODS THE VALUE
C  IS TO BE SPREAD OVER.
 1010 CALL DBLK(A,B,NA,NB,IB,NFILE,IEOF)
      IF (IEOF .NE. 1) GO TO 1600
C     DETERMINE WHAT TO DO WITH THESE STATISTICS
 1050 IF (ITX - ITXTF) 1060,1080,1010
C
 1060 READ (2) ITX,NORFI,TYPSAT,
     *                 ((IEXTX(J,I),J=1,5),I=1,NORFI)
      IF (EOF,2) 1600,1050
C
 1080 DO 1099 I=1,NORFI
      TORP = TNOW
      AAAA = A(4)
      IF (TYPSAT .EQ. 1.0) TORP=TORP-A(4)
      IS = IEXTX(2,I)
      JCODE = MOD(ISELCT(IS+3),10)
 1081 JT = 1.0+(TORP-SELCT(IS))/SELCT(IS+1)
      IF (TYPSAT .EQ. 1.0) AAAA=AMIN1(A(4),SELCT(IS)+SELCT(IS+1)*
     *                   FLOAT(JT)-TORP, TNOW-TORP)
 1082 IF (JT .GE. 1) GO TO 1083
      IF (TYPSAT .NE. 1.0) GO TO 1099
      TORP = SELCT(IS)+FLOAT(JT)*SELCT(IS+1)
      IF (TORP .GE. TNOW) GO TO 1099
      GO TO 1081
 1083 IF (JT .GT. ISELCT(IS+2)) GO TO 1099
C     PASSED THE TIME CHECK.
C
      IF (IEXTX(3,I) .EQ. 0) GO TO 1086
C     VARIABLE SET ONE.
      IF (ITX .EQ. LITX .AND. JT .EQ.  IT) GO TO 1087
      IF (LITX .NE. 0) GO TO 1085
 1084 LITX = ITX
      IT = 1
      TCODE = JCODE
      ISP = IEXTX(2,I)
      ISV = IEXTX(3,I)
```

```
      ISN = IEXTX(4,I)
      ISR = IEXTX(5,I)
      IF (JT .EQ. 1) GO TO 1087
 1085 CALL LABALL(3)
      IF (ITX .EQ. LITX) GO TO 1089
      CALL LABALL(4)
      GO TO 1084
 1089 IT = IT+1
      IF (IT .LT. JT) GO TO 1085
      IF (IT.EQ.JT) GO TO 1087
      WRITE(6,2010) A,JT,IT
 2010 FORMAT(34H TRAFAN ENCOUNTERED A TRANSACTION , 4F12.3,
     *   23H TO BE ENTERED IN T.P.  ,I5,17H NOW ACCUM. T.P.  ,I5,
     *   /26H ABOVE TRANSACTION IGNORED   )
 1087 IXX = 1
      GO TO 1088
 1086 IXX = IEXTX(1,I)+JT-1
 1088 GO TO(1090,1090,1092,1094), JCODE
C     COUNT.
 1090 XCOUNT(1) = A(3)
      XCOUNT(2) = AA44
      CALL COUNT(JCODE)
      GO TO 1096
C     COLLECT.
 1092 XCOLCT = A(3)
      WTCLCT = AA44
      CALL COLCT
      GO TO 1096
C     HISTOGRAM.
 1094 XHISTO = A(3)
      WTHSTO = AA44
      XLO = SELCT(IS+4)
      W = SELCT(IS+5)
      CALL HISTO
C  CHECK IF TIME PERIOD JT IS UP TO T.P. KT, THAT IS, IF THE CURRENT
C  VALUE SHOULD BE PLACED IN ANOTHER CELL BEFORE NEXT TRANSACTION
C  IS SELECTED
 1096 TORP = TORP+SELCT(IS+1)
      AA44=AMIN1(SELCT(IS+1),TNOW-SELCT(IS+1)*FLOAT(JT)-SELCT(IS))
      JT = JT+1
      IF (TORP .GT. TNOW) TORP = TNOW
      KT = 1.0+(TORP-SELCT(IS))/SELCT(IS+1)
      IF (KT.GE.JT) GO TO 1082
 1099 CONTINUE
      GO TO 1010
C     END OF EXTRACT.
 1600 IF (ISV .EQ. 0) GO TO 2000
      CALL LABALL(3)
      CALL LABALL(4)
 2000 CALL LABALL(2)
      CALL ENDB(GB,LGB,LGA,IGB,NGRPHF)
      REWIND NFILE
      REWIND NGRPHF
      END FILE 8
      REWIND 8
      STOP
      END
```

F. PROGRAM PGRAPH

The purpose of program PGRAPH is to read the Graph File and to organize the data such that the subroutine GRAPH can print the graphs. PGRAPH has been altered to add the report or reports number for each cross variable in the graph. This program utilizes the report numbers that have been added to the graph file by program PRNTSO.

```
      PROGRAM PGRAPH (INPUT,OUTPUT,TAPE6=OUTPUT,TAPE4)
C
C     THIS PROGRAM READS A GRAPH FILE AS PREPARED BY TRAFAN
C     AND GENERATES THE ARRAYS AS REQUIRED BY THE SUBROUTINE
C     GRAPH WHICH GENERATES GRAPHS ON THE COMPUTER PRINTER.
C
      COMMON /GSPDUM/ TBEG,TFIN,NCLCT,MON,NDAY,NYR,RUNNAM(3),
     *                MODNAM(2),NAME(2),NMDPP(5)
      DATA BLANK /1H /
      DIMENSION A(500,9),B(500),TITLE(9),BT(3),CT(6,9),
     *          GA(3), IGA(3), GB(300), FT(6), IFT(6)
      EQUIVALENCE (GA,IGA), (FT,IFT)
      DATA FT /6H(     ,6H4H0GRA,6H4HPH N,6H4H0,,,,6HI4,5X,,6H8A6 ) /
      DATA  REPRTS,PLUS   /6HREPRTS,6H  +...  /
      INTEGER RUNNAM, TITLE
      LT = 9
      MAXNVV = 500
C     INITIALIZE GRAPH FILE ARRAYS.
      NGRPHF = 4
      MODE = 2
      LGA = 3
      LGB = 300
      CALL INITB (GB,LGB,IGB,NGRPHF,MODE,IEOFG)
C     STORE TITLE DATA.
      DO 116 I=1,9
      CALL DBLK(GA,GB,LGA,LGB,IGB,NGRPHF,IEOFG)
      GO TO (100,102,104,106,108,110,112,114,116), I
100   MON  = IGA(2)
      NDAY = IGA(3)
      GO TO 116
102   NYR        = IGA(2)
      RUNNAM(1) = IGA(3)
      GO TO 116
104   RUNNAM(2) = IGA(2)
      RUNNAM(3) = IGA(3)
      GO TO 116
106   MODNAM(1) = IGA(2)
      MODNAM(2) = IGA(3)
      GO TO 116
108   NAME(1) = IGA(2)
      NAME(2) = IGA(3)
      GO TO 116
110   NMDPP(1) = IGA(2)
      NMDPP(2) = IGA(3)
      GO TO 116
112   NMDPP(3) = IGA(2)
      NMDPP(4) = IGA(3)
      GO TO 116
114   NMDPP(5) = IGA(2)
116   CONTINUE
C
      CALL PGHDRD
C     INITTALIZE THE ARRAYS AND INDICIES
200   NOCV = 0
      NOHV = 0
      NOG = IGA(1)/1000
      TITLE(1) = NOG
```

```
      DO 202 I=2,8,2
      TITLE(I)   = IGA(2)
      TITLE(I+1) = IGA(3)
      CALL DBLK(GA,GB,LGA,LGB,IGB,NGRPHF,IEOFG)
      IF (IEOFG .NE. 1) GO TO 212
202   CONTINUE
      DO 206 I=1,6
      DO 204 J=1,9
204   CT(I,J) = BLANK
206   BT(I) = BLANK
C
207   IF (NOG .NE. IGA(1)/1000) GO TO 200
      I = MOD((IGA(1)/100),10)
      IF (I .EQ. 3) GO TO 214
      IF (I .EQ. 2) GO TO 208
C     BASE VARIABLE TITLE.
      BM = GA(2)
      BT(1) = GA(3)
      CALL DBLK(GA,GB,LGA,LGB,IGB,NGRPHF,IEOFG)
      BT(2) = GA(2)
      BT(3) = GA(3)
      GO TO 210
C     CROSS VARIABLE TITLE.
208   J = MOD(IGA(1)/10,10)
      IF (MOD(IGA(1),10).EQ.3) GO TO 209
      IF (J .EQ. 1) CM = GA(2)
      IF (NOCV .LT. J) NOCV = J
      CT (1,J) = GA(3)
      CALL DBLK(GA,GB,LGA,LGB,IGB,NGRPHF,IEOFG)
      CT(2,J) = GA(2)
      CT(3,J) = GA(3)
      GO TO 210
C     REPORT NUMBER FOR CROSS VARIBLE     1   OR 1   2  OR  1  +...
209   IF (CT(4,J) .NE. BLANK) GO TO 9209
      CT(4,J) = GA(2)
      CT(5,J) = GA(3)
      GO TO 210
9209  IF (CT(4,J) .EQ. REPRTS) GO TO 9219
      CT(4,J) = REPRTS
      CT(6,J) = GA(3)
      GO TO 210
9219  CT(6,J) = PLUS
210   CALL DBLK(GA,GB,LGA,LGB,IGB,NGRPHF,IEOFG)
211   IF (IEOFG.EQ.1) GO TO 207
212   CALL PGHDRO
      STOP
C
C
214   DO 216 I=1,MAXBVV
      B(I) = 0
      DO 216 J=1,NOCV
216   A(I,J) = 0
      IGCK = NOG*1000+400
      IFG1 = 0
C     STORE VALUES OF GRAPH.
220   IF (IGA (1) .GT. IGCK) GO TO 250
      J = MOD(IGA(1)/10,10)
```

```
C       FIND ROW TO STORE.
        IF (J .GT. 1) GO TO 226
        IF (NOBV .EQ. 0) GO TO 222
        IF (B(NOBV) .EQ. GA(2)) GO TO 224
222     IF (NOBV .EQ. MAXBVV) GO TO 244
        NOBV = NOBV+1
        B(NOBV) = GA(2)
224     A(NOBV,1) = A(NOBV,1)+GA(3)
        GO TO 244
226     IF (IFG1 .NE. 0) GO TO 230
        IFG1 = 1
        NOSBV = NOBV
        IF (NOBV .EQ. 0) GO TO 228
        NOCKS =  INT(ALOG(FLOAT(NOBV))/ALOG(2.0))
        MID = 2**NOCKS
        NOCKS = NOCKS+1
        GO TO 230
228     NOCKS = 0
230     IF (NOCKS .EQ. 0) GO TO 237
        II = MID
        IJ = MID
        DO 236 I=1,NOCKS
        IJ = IJ/2
        IF (II .GT. NOSBV) GO TO 232
        IF (GA(2) - B(II)) 232,242,234
232     II = II-IJ
        GO TO 236
234     II = II+IJ
236     CONTINUE
C
237     II = NOSBV+1
238     IF (II .GT. NOBV) GO TO 240
        IF (GA(2) .EQ. B(II)) GO TO 242
        II = II+1
        GO TO 238
C
240     IF (II   .GT. MAXBVV) GO TO 244
        B(II) = GA(2)
        NOBV = II
242     A(II,J) = A(II,J)+GA(3)
C
244     CALL DBLK(GA,GB,LGA,LGB,IGB,NGRPHF,IEOFG)
        IF (IEOFG .EQ. 1) GO TO 220
C
250     IF (NOCV .EQ. 0  .OR. NOBV .EQ. 0) GO TO 260
C
        CALL REARG(A,MAXBVV,9,A,NOBV,NOCV)
C       SORT ORDER
        NS = 0
        IF (NOBV .GT. NOSBV) NS = 2
C       LIMITS
        CU = 0
        CL = 0
C       ORIENTATION
        IO = 1
        IF (NOBV .GT. 101) IO = 0
C       CONTINUOUS PAGE.
```

```
      IP = 0
C     NUMBER OF LINES.
      NL = 48
      CALL PGHDRO
      CALL GRAPH (NOBV,NOCV,A,B,NL,NS,CU,CL,IO,IP,TITLE,LT,FT,BT,CT,
     *            BM,CM)
C
260   IF (IEOFG .EQ. 1) GO TO 200
C
      CALL PGHDRO
      REWIND NGRPHF
      STOP
      END
```

G. SUBROUTINE ENDB

ENDB is one of the set of four subroutines used to read and write binary files (see subroutine BLOCK). The purpose of ENDB is to fill the last block with records that will fall at the end or a sorted file, write the block out, and write an end of file mark. These records have a constant of 10^{10} in the first word and the negative value of that constant in the second word, to sort properly on ascending, (word one) and descending, (word two) sorts. The calling sequence is

ENDB (B, NB, IB, NFILE)

where

B is the block array,

NB is the length of the array B,

IB is the next position in which data are to be stored,

NFILE is the logical number of the file.

This subroutine must be used to close each file that is written using BLOCK.

```
      SUBROUTINE ENDB(B,NB,NA,IB,NFILE)
C ENDS FILE NFILE BY FILLING B WITH NINES, WRITING THIS OUT, AND WRITIN
C  END OF FILE.
       DIMENSION B(NB)
      DATA FILL /1.E10/
      NANOW = NA
      DO 10 I=IB,NB,NANOW
      B(I) = FILL
      IF ((I+1) .GT. NB) GO TO 10
      B(I+1) = -FILL
   10 CONTINUE
C WRITE
       WRITE (NFILE) B
       END FILE NFILE
       IB=0
       RETURN
       END
```

H. SUBROUTINE PRNTSO

The purpose of the subroutine PRNTSO is to have the print line output properly according to the type of statistics collection, to have the report title printed where necessary, and to write the data onto the Graph File and the Cost File. The calling sequence is

PRNTSO (TYPIND, IT, T)

where

TYPIND was the collection type code and storage index in compound form,

IT is the time period number,

T is the time at the end of the time period.

Program PRNTSO now adds extra records to the Graph File to store the report number for each cross variable in a graph. These records have a key word with value 203 to enforce them to fall in proper sequence when the Graph File is sorted.

```
      SUBROUTINE PRNTSO(TYPIND,IT,T,IPGRPH)
C
C           TYPE CODE
C               1 - COUNT - OF ARGUMENT X1IN
C               2 - COUNT - OF ARGUMENT X2IN
C               3 - COLCT
C               4 - HISTO
C               5 - TACTN
C
C     THE LINE COUNTER IS INCREMENTED FOR ALL PRINT LINES PRINTED BY
C     STAT.  LINES FOR SECTION HEADERS PRINTED BY THE CALLING ROUTINE
C     SHOULD BE COUNTED IN THAT ROUTINE
C
      COMMON /GSPDUM/ TBEG,TFIN,NCLCT,MON,NDAY,NYR,RUNNAM(3),
     *                MODNAM(2),NAME(2),NMDPP(5)
      COMMON /AR/ NPRNT
      DATA NPRNT /6/
      COMMON /LINCOM/LINE,IPGSKP,ICLHR
      COMMON /AGENDA/ IX(1000), ISELCT(3000), NOSVS, IXMAX, ISMAX
      DATA IXMAX, ISMAX /1000,3000/
      COMMON /SVHSTO/ XLO, A, H(22, 320), NH( 320), NHMAX,
     *                NHISTO,NXTHST,F(22),CF(22),AVG,SD,MI
      DATA NHISTO, NHMAX / 320, 22/
      COMMON /GRPHDA/ IGRAPH(6,400), GB(300), LIG, LGB, LGA,
     *                LINDXG, IGB, NGRPHF, IEOFG, GA(3)
      DATA LIG, LGH, LGA, NGRPHF / 400,300,3,3/
      COMMON /IOR/ KV,KN1,KN2,KR1,KR2,MSG,NRPT,KG1,KG2,ITYP
C
      EQUIVALENCE (GA,IG)
      DIMENSION OV(6)
      DIMENSION ANMBR(11)
      DATA ANMBR /1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,1H    /
      DATA BLANK /1H /, REPORT /6HREPORT /
      DATA TO / 4 /
C     DETERMINE TYPE AND INDEX
C
      INDEX = TYPIND/10.
      IF (ITYP .LE. 0 .OR. ITYP .GT. 5) RETURN
      IF(INDEX.GT.0) GO TO 100
C     NO ACTIVITY
      RETURN
  100 GO TO (1,1,1,4,5),ITYP
C
C     COUNT TYPE STATISTIC
    1 IF (IPGSKP .EQ. 0 .AND. IT .NE. 1) GO TO 10
      IF (LINE+5 .GE. 60) CALL LIN(6)
      CALL REPTTL
      CALL CLTHRO
      CALL LIN(2)
      IF (IT.NE.1) GO TO 10
C  STORE REPORT NUMBER ON GRAPH FILE (KEY = 203)
      IF (KG1.LE.0) GO TO 10
C CONVERT NRPT VALUE TO ALPHA
      NTEMP = NRPT
      I3 = MOD(NTEMP,10)
      NTEMP = NTEMP/10
      I2 = MOD(NTEMP,10)
```

```
      I1 = NTEMP/10
C  LEADING ZEROS DELETED
      IF (NRPT.LT.100) I1 = 10
      IF (NRPT.LT.10) I2 = 10
C  PACK ALPHA INTEGERS INTO ATEMP
      ENCODE(4,1008,ATEMP) BLANK,ANMBR(I1+1),ANMBR(I2+1),ANMBR(I3+1)
 1008 FORMAT(4A1)
      DO 8 KG=KG1,KG2
      IF (IGRAPH(3,KG).NE.0 .AND. IGRAPH(3,KG).NE.KV  ) GO TO 8
      IF (IGRAPH(4,KG).NE.0 .AND. IGRAPH(4,KG).NE.ISELCT(KN1)) GO TO 8
      IF (IGRAPH(5,KG).NE.0 .AND. IGRAPH(5,KG).NE.ISELCT(KR1)) GO TO 8
      IG = IGRAPH(1,KG)-98
      GA(2) = REPORT
      GA(3) = ATEMP
      CALL BLOCK(GA,GB,LGA,LGB,IGB,NGRPHF)
    8 CONTINUE
   10 GO TO (11,11,13), ITYP
   11 CALL CNTLNO (INDEX,IT,T,OV)
      OV(2) = OV(1)
      OV(1) = 0
      OV(3) = 0
      OV(4) = 0
      WRITE (8) NRPT,IT,I4,(OV(I),I=1,I4)
C     COUNT GRAPH.
      IF(IPGRPH.EQ.0) GO TO 12
      GA(2) = IT
      IF (KG1 .EQ. 0) GO TO 12
      DO  9 KG=KG1,KG2
      IF (IGRAPH(3,KG) .NE.    0 .AND. IGRAPH(3,KG) .NE. KV) GO TO 9
      IF (IGRAPH(4,KG) .NE.    0 .AND. IGRAPH(4,KG) .NE. ISELCT(KN1))
     *   GO TO 9
      IF (IGRAPH(5,KG) .NE.    0 .AND. IGRAPH(5,KG) .NE. ISELCT(KR1))
     *   GO TO 9
      IG = IGRAPH(1,KG)
      GA(3) = OV(2)
      CALL BLOCK(GA,GB,LGA,LGB,IGB,NGRPHF)
    9 CONTINUE
      GO TO 12
C
C     COLCT TYPE STATISTIC
   13 CALL CLTLNO (INDEX,IT,T,OV)
      WRITE (8) NRPT,IT,I4,(OV(I),I=1,I4)
C     COLLECT GRAPH.
      IF(IPGRPH.EQ.0) GO TO 12
      IF (KG1 .EQ. 0) GO TO 12
      GA(2) = IT
      DO 14 KG=KG1,KG2
      IF (IGRAPH(3,KG).NE.0.AND.IGRAPH(3,KG).NE.KV) GO TO 14
      IF (IGRAPH(4,KG).NE.0.AND.IGRAPH(4,KG).NE.ISELCT(KN1)) GO TO 14
      IF (IGRAPH(5,KG).NE.0.AND.IGRAPH(5,KG).NE.ISELCT(KR1)) GO TO 14
      IGS = IGRAPH(2,KG)
      IG = IGRAPH(1,KG)
      GA(3) = OV(IGS)
      CALL BLOCK(GA,GB,LGA,LGB,IGB,NGRPHF)
   14 CONTINUE
   12 CALL LIN(1)
      RETURN
```

```
C
C       HISTOGRAM TYPE STATISTICS COLLECTION
      4 IF(NH(INDEX).GT.0) GO TO 40
C       EMPTY HISTOGRAM
        IF (IT .EQ. 1 .AND. LINE+5  .GE. 60) LINE=61
        IF (IT .NE. 1 .AND. LINE+1 .GE. 60) LINE=61
        CALL LIN(0)
        IF (IT .EQ. 1 .OR. IPGSKP .NE. 0) CALL REPTTL
        WRITE (NPRNT, 43) IT
        CALL LIN(1)
     43 FORMAT (21X,33HNO HISTOGRAM DATA FOR TIME PERIOD,I6)
        RETURN
C
C       PRINT HISTOGRAM
     40 LINC = (NH(INDEX)+1)/2 + 7
        IF (IPGSKP .EQ. 1) GO TO 41
        IF (IT .EQ. 1 .AND. LINE+LINC+4 .GE. 60) LINE =61
        IF (IT .NE. 1 .AND. LINE+LINC .GE. 60) LINE = 61
        CALL LIN(0)
        IF (IT .EQ. 1) GO TO 41
        IF (IPGSKP .EQ.0) GO TO 42
     41 CALL REPTTL
     42 CALL HRPTO (INDEX,IT,T,OV)
        WRITE (8) NRPT,IT,I4,(OV(I),I=1,I4)
        CALL LIN(LINC)
C       HISTOGRAM GRAPH.
        IF(IPGRPH.EQ.0) RETURN
        IF (KG1 .EQ. 0) RETURN
        DO 50 KG=KG1,KG2
        IF (IGRAPH(3,KG).NE.0.AND.IGRAPH(3,KG).NE.KV) GO TO 50
        IF (IGRAPH(4,KG).NE.0.AND.IGRAPH(4,KG).NE.ISELCT(KN1)) GO TO 50
        IF (IGRAPH(5,KG).NE.0.AND.IGRAPH(5,KG).NE.ISELCT(KR1)) GO TO 50
        IGS = IGRAPH(2,KG)
        IG = IGRAPH(1,KG)
        IF (IGS .GT. 6) GO TO 44
        GA(2) = IT
        GA(3) = OV(IGS)
        CALL BLOCK(GA,GB,LGA,LGB,IGB,NGRPHF)
        GO TO 50
C       INTERVAL CASE.
     44 NN = NH(INDEX)
        IF (IGRAPH(6,KG).NE. IT) GO TO 50
        GA(2) = XLO-2*W
        DO 48 I=1,NN
        GA(2) =GA(2)+W
        IF (IGS-8) 45,46,47
     45 GA(3) = H(I,INDEX)
        GO TO 48
     46 GA(3) = F(I)
        GO TO 48
     47 GA(3) = CF(I)
     48 CALL BLOCK(GA,GB,LGA,LGB,IGB,NGRPHF)
     50 CONTINUE
      5 RETURN
        END
```

1. SUBROUTINE REPTTL

The purpose of the subroutine REPTTL is to print the report title data including the definition of the transaction of the statistics that follow. Each report that is a continuation at the beginning of a page is so marked. There are no parameters in the calling sequence. All data must be stored in the COMMON areas.

The references to the graph and cross variable number in the report title have been improved in this version of REPTTL.

```
      SUBROUTINE REPTTL
C
      COMMON /WR/ NPRNT
      DATA NPRNT /6/
      COMMON /AGENDA/ IX(1000), ISELCT(3000), NOSVS, IXMAX, ISMAX
      DATA IXMAX, ISMAX /1000,3000/
      COMMON /SEQI/ISV,ISN,MAXN,IGON,ISR,MAXR,IGOR,IT,ISP,ICODE,IV
     *              ,IN,IR,LENN,LENR,LENT
      COMMON /TOR/ KV,KN1,KN2,KR1,KR2,MSG,NRPT,KG1,KG2,ITYP
      COMMON /GRPHDA/ IGRAPH(6,400), GB(300), LIG, LGB, LGA,
     *                LINDXG, IGR, NGRPHF, IEOFG, GA(3)
      DATA LIG, LGB, LGA, NGRPHF / 400,300,3,3/
      DATA IO / 0 /
    1 FORMAT (10H0***** NO.,I5,3X,7A6,13X,20H(TRANSACTION TYPE...,I10,
     *        1H),12X,2A6,1X,5H*****)
    2 FORMAT (6H      ,15X,9HNODES....,9I11)
    3 FORMAT (6H      ,15X,9HNODES....,8X,3HALL)
    4 FORMAT (21X,9HRESOURCES,9I11)
    5 FORMAT (21X,9HRESOURCES,8X,3HALL)
    6 FORMAT (21X,9HRESOURCES,2X,7HPRIMARY,I13,22H, WITH ALL SECONDARIES
     *       )
    7 FORMAT (21X,9HRESOURCES,2X,9HSECONDARY,I11,20H, WITH ALL PRIMARIES
     *       )
    8 FORMAT(6H      ,15X,12HGRAPH NO....,I8,5X,18HCROSS VARIABLE.... ,
     1         I2)
      DIMENSION CONT(2), OUT(2)
      DATA CONT,BLK/6HCONTIN,6HUATION,1H /
      DO 12 I=1,2
      IF (IT.EQ.1) GO TO 11
      OUT(I)=CONT(I)
      GO TO 12
   11 OUT(I)=BLK
   12 CONTINUE
C
      IEND=MSG+6
      WRITE (NPRNT,1)  NRPT, (ISELCT(I),I=MSG,IEND), KV, OUT
C  GRAPH IDENTIFICATION LINE.
      IF (KG1.LE.0) GO TO 112
      DO 111 KG = KG1,KG2
      IF (IGRAPH(3,KG).NE.0 .AND. IGRAPH(3,KG).NE.KV)        GO TO 111
      IF (IGRAPH(4,KG).NE.0 .AND. IGRAPH(4,KG).NE.ISELCT(KN1)) GO TO 111
      IF (IGRAPH(5,KG).NE.0 .AND. IGRAPH(5,KG).NE.ISELCT(KR1)) GO TO 111
      NOG = IGRAPH(1,KG )/1000
      NCV = MOD(IGRAPH(1,KG )/10,10)
      WRITE(NPRNT,8) NOG,NCV
      GO TO 112
  111 CONTINUE
  112 CONTINUE
C     NODE LINE.
      GO TO (13,13,14), IGON
   13 WRITE (NPRNT,2) (ISELCT(I),I=KN1,KN2)
      NND = KN2-KN1+1
      GO TO 15
   14 WRITE (NPRNT,3)
      NND = 0
C     RESOURCE LINE.
   15 GO TO (16,16,17,18,19), IGOR
```

```
16 WRITE (NPRNT,4) (ISELCT(I),I=KR1,KR2)
   NR1 = KR1
   NR2 = KR2
   NRS = NR2-NR1+1
   GO TO 20
17 WRITE (NPRNT,5)
   NRS = 0
   NR1 = 1
   NR2 = 1
   GO TO 20
18 WRITE (NPRNT,6) ISELCT(ISR)
   NRS = -1
   NR1 = ISR
   NR2 = ISR
   GO TO 20
19 WRITE (NPRNT,7) ISELCT(ISR)
   NRS = -2
   NR1 = ISR
   NR2 = ISR
20 CALL LIN(4)
   IF (IT .NE. 1) RETURN
   L = 4+MAX0(0,NND)+MAX0(0,NRS)
   IF (NRS .LT. 0) L=L+1
   IF (NND .NE. 0) WRITE (8) NRPT,IO,L,ITYP,KV,NND,NRS,
  *                  (ISELCT(I),I=KN1,KN2),(ISELCT(I),I=NR1,NR2)
   IF (NND .EQ. 0) WRITE (8) NRPT,IO,L,ITYP,KV,NND,NRS,
  *                                        (ISELCT(I),I=NR1,NR2)
   RETURN
   END
```

J. SUBROUTINE GRAPH

The program GRAPH prints out the graph from the array values set up by program PGRAPH, properly scaling the horizontal and vertical axis for the values to be plotted. The title section for cross variable identification has been altered to provide two extra lines for the report number or numbers from which the points are plotted.

```
      SUBROUTINE GRAPH (M,N,A,B,NL,NS,CU,CL,IO,IP,T,LT,FT,BT,CT,BM,CM)
C
C  PURPOSE
C     PLOT UP TO NINE CROSS VARIABLES VERSUS A BASE VARIABLE WITH THE
C     BASE VARIABLE PLACED EITHER ON THE SIDE OR THE BOTTOM OF THE PAGE.
C
C  DESCRIPTION OF PARAMETERS
C     M  - NUMBER OF BASE VARIABLE VALUES.
C     N  - NUMBER OF CROSS VARIABLES.
C     A  - MATRIX OF DATA TO BE PLOTTED.  A(I,J) HAS THE VALUE OF THE
C          J-TH CROSS VARIABLE AT THE I-TH BASE VARIABLE VALUE.
C     B  - MATRIX OF BASE VARIABLE VALUES.
C     NL - NUMBER OF LINES.  IF NL IS LESS THAN 1 THEN NORMAL OF 46.
C     NS - SORT ORDER OF BASE VARIABLES.  0- IN ASCENDING ORDER, 1- IN
C          DESCENDING ORDER, 2- SORT INTO ASCENDING ORDER, 3- SORT INTO
C          DESCENDING ORDER.
C     CU - CROSS VARIABLE UPPER LIMIT.
C     CL - CROSS VARIABLE LOWER LIMIT.  IF CU=CL=0 THEN MAKE MAX AND MIN
C     IO - ORIENTATION.  0- BASE ON SIDE, 1- BASE ON BOTTOM.
C     IP - CONTINUOUS PAGE FLAG.  0- CONTINUOUS, 1- PAGE BREAK (47 GRAPH
C          LINES ON FIRST PAGE AND UP TO 50 GRAPH LINES ON OTHER PAGES.)
C     T  - TITLE ARRAY.
C     LT - LENGTH OF TITLE ARRAY.
C     FT - FORMAT OF TITLE ARRAY. (ASSUMED TO BE ONE LINE).
C     BT - BASE VARIABLE TITLE (BT(3), 3A6).
C     CT - CROSS VARIABLE TITLE (CT(3,N), 3A6 EACH).
C     BM - UNITS OF MEASURE OF BASE VARIABLE VALUES.
C     CM - UNITS OF MEASURE OF CROSS VARIABLE VALUES.
C
      DIMENSION A(M,N),B(M),BT(3),CT(6,N),T(LT)
C
      DIMENSION OC(101),SYM(10),CVL(3,40),ST(40),BA(4),TMS(3),BL(11)
C
      DATA SYM/6H******,6H++++++,6H&&&&&&,6H------,6H======,6H$$$$$$,
     *         6H000000,6HXXXXXX,6H??????,6H||||||/
      DATA TMS,BLANK/6HMULTIP,6HLE VAR,6HIABLES,6H      /
C     TOP LINE.
1     FORMAT (1H0,15HCROSS VARIABLES,13X,3H++-,10(10H---------+-))
C     MIDDLE LINES.
2     FORMAT (1X,3A6,2X,A1,1X,F6.0,1H+,101A1,1H+)
3     FORMAT (1X,3A6,2X,A1,7X,1HI,101A1,1HI)
C     BOTTOM LINES.
4     FORMAT (1X,18HVARIABLE X FACTORS,10X,3H-+-,10(10H---------+-)/
     *        1X,10HBASE.....,E8.1, 2X,11(4X,F6.0)/
     *        1X,10HCROSS.....,E8.1,49X,3A6,2X,A6)
C     PAGE BREAK.
5     FORMAT (1H1)
      M=M+N
C     CHECK SORT.
      IF (NS.LT.2) GO TO 50
      IF (NS.EQ.2) GO TO 15
      GO TO [illegible]
```

```
      K=I
      J=I+L
      IF (J.GT.M) GO TO 40
25    IF (B(K).LE.B(J)) GO TO 35
      C=B(K)
      B(K)=B(J)
      B(J)=C
      DO 30 N1=1,N
      C=A(K,N1)
      A(K,N1)=A(J,N1)
30    A(J,N1)=C
      J=K
      K=K-L
      IF (K.GT.0) GO TO 25
35    CONTINUE
40    IF (L.GT.1) GO TO 20
      IF (NS.EQ.2) GO TO 50
      DO 45 I=1,M
45    B(I)=-B(I)
C     PRINT TITLE,
50    WRITE (6,FT) (T(I),I=1,LT)
C
      CUL=CU
      CLL=CL
      IF (CU.NE.0.0 .OR. CL .NE.0.0) GO TO 60
      CUL=A(1,1)
      CLL=A(1,1)
      DO 55 J=1,N
      DO 55 I=1,M
      IF (CUL.LT.A(I,J)) CUL=A(I,J)
      IF (CLL.GT.A(I,J)) CLL=A(I,J)
55    CONTINUE
      IF (CUL .NE. CLL) GO TO 60
      CLL = CLL-10.0
      CUL = CUL+10.0
C
60    IF (NL.GT.500) GO TO 410
      NLL=NL
      IF (NL.LT.1) NLL=46
C
      DO 65 J=1,40
      ST(J)=BLANK
      DO 65 I=1,3
65    CVL(I,J)=BLANK
      CVL(2,2)=SYM(10)
      CVL(1,3)=TMS(1)
      CVL(2,3)=TMS(2)
      CVL(3,3)=TMS(3)
      DO 72 I=1,N
      L = 4*I+2
      CVL(2,L)=SYM(I)
      L=L+1
      DO 70 J=1,3
70    CVL(J,L)=CT(J,I)
      L = L+1
      DO 72 J=4,6
72    CVL(J-3,L) = CT(J,I)
```

```
C
      IF (NLL.LT.4*N+4) NLL = 4*N+4
C
      IF (ID.EQ.1) GO TO 200
C     BASE ON SIDE.
      NLL=MAX0(M,NLL)
      FNLL=NLL
      C=B(2)-B(1)
      DO 75 I=3,M
      IF (B(I)-B(I-1).NE.C) GO TO 85
75    CONTINUE
C     EQUAL INCREMENTS.
      H=INT((FNLL-1.0)/FLOAT(M-1))
      DS=C/H
      IL=INT((DS*(FNLL-1.0)-C*FLOAT(M-1))/(2.0*DS))+1
      BS=B(1)-DS*FLOAT(IL-1)
      IH=H
      IHI=MOD(IL,IH)
      C=B(1)
      D=B(M)
      DO 80 I=1,M
80    B(I)=IL+IH*(I-1)
      GO TO 95
C     UNEQUAL INCREMENTS.
85    DS=(B(M)-B(1))/(FNLL-1.0)
      H=4
      IH=H
      IHI=1
      BS=B(1)
      C=B(1)
      D=B(M)
      DO 90 I=1,M
90    B(I)=AINT((B(I)-BS+.5*DS)/DS)+1.0
C     SCALING OF BASE VARIABLE.
95    H=DS*H
      CALL SCALE(C,D,H,BSF)
      BS=BS/BSF
      DS=DS/BSF
C     CROSS ON BOTTOM.
125   DC=(CUL-CLL)/100.0
      C=1.0/DC
      DO 130 I=1,M
      L=B(I)*10000.0
      DO 130 J=1,N
      K=((A(I,J)-CLL)*C+.5)+1.0
      IF (K.LT.1 .OR. K.GT.101) K=0
130   A(I,J)=L+K*10+J
C     SCALING OF CROSS VARIABLE.
      H=DC*H
      CALL SCALE(CUL,CLL,H,CSF)
      DO 155 J=1,11
155   BL(J)=(CLL+DC*FLOAT(J*10-10))/CSF
C     STORE LABLES.
      DO 165 J=1,4
165   BA(J)=BLANK
      BA(2)=CM
      I=5
```

```
      IF (NLL.LT.30) I=2
      J=I+23
C     FOR COMPUTERS WITHOUT DECODE USE THE FOLLOWING WRITE-READ SEQ.
C     REWIND 10
C     WRITE (10,170) BT,BM
C170  FORMAT (4A6)
C     REWIND 10
C     READ (10,175) (ST(K),K=I,J)
C175  FORMAT (24A1)
      J=I+17
      DECODE (30,180,BT) (ST(K),K=I,J)
      I=J+3
      J=J+8
      DECODE (6,180,BM) (ST(K),K=I,J)
180   FORMAT (4(6A1,4X))
      GO TO 306
C
C     BASE ON BOTTOM.
200   DO 205 J=1,3
205   BA(J)=BT(J)
      BA(4)=BM
      I=NLL/2
      IF (I.GT.25) I=25
      IF (I.LT.2) I=2
      J=I+5
C     REWIND 10
C     WRITE (10,210) CM
C210  FORMAT (A6)
C     REWIND 10
C     READ (10,215) (ST(K),K=I,J)
C215  FORMAT (6A1)
      DECODE (6,180,CM) (ST(K),K=I,J)
C
      C=B(2)-B(1)
      DO 225 I=3,M
      IF (B(I)-B(I-1).NE.C) GO TO 240
225   CONTINUE
      IF (M.GT.101) GO TO 420
      D=B(1)
      H=C
      C=AINT(100.0/FLOAT(M-1))
      B(1)=AINT(.5*(100.0-C*FLOAT(M-1)))+1.0
      DO 230 I=2,M
230   B(I)=B(I-1)+C
      C=H/C
      DO 235 I=1,11
235   BL(I)=D+C*(FLOAT(I-1)*10.0+1.0-B(1))
      GO TO 255
C     UNEQUAL INCREMENTS.
240   C=(B(M)-B(1))/100.0
      D=B(1)
      DO 245 I=1,M
245   B(I)=AINT((B(I)-D)/C)+1.0
      DO 250 I=1,11
250   BL(I)=D+FLOAT(I*10-10)*C
C     SCALE BASE.
255   H=C*10.0
```

```
      CALL SCALE(BL(1),BL(10),H,BSF)
      IF (BSF.EQ.1) GO TO 300
      DO 295 I=1,11
295   BL(I)=BL(I)/BSF
C     CROSS VARIABLES.
300   FNLL=NLL
      DC=(CUL-CLL)/(FNLL-1.0)
      IF (DC .EQ. 0.0) DC = 1.0
      C=1.0/DC
      DO 305 I=1,M
      L=B(I)*10.0
      DO 305 J=1,N
      K=NLL-INT((A(I,J)-CLL)*C+.5)
      IF (K.LT.1 .OR. K.GT.NLL) K=0
305   A(I,J)=K*10000+L+J
C     SCALING OF CROSS.
      H=DC*5.0
      CALL SCALE(CLL,CUL,H,CSF)
      BS=CUL/CSF
      DS=-DC/CSF
      IH=5
      IHI=1
C     SORT LCA.
306   L=MN
      IF (L.LT.1) GO TO 400
310   L=(L+1)/2
      DO 320 I=1,MN
      K=I
      J=I+L
      IF (J.GT.MN) GO TO 325
315   IF (A(K).LE.A(J)) GO TO 320
      C=A(K)
      A(K)=A(J)
      A(J)=C
      I=K
      K=K-L
      IF (K.GT.0) GO TO 315
320   CONTINUE
325   IF (L.GT.1) GO TO 310
C     PRINT THE REPORT.
330   K=1
      LINE=3
      WRITE (6,1)
      DO 390 I=1,NLL
      C=FLOAT(I)*10000.0
      DO 335 J=1,101
335   OC(J)=BLANK
      IF (K.GT.MN) GO TO 370
340   L=A(K)/10000.0
345   IF (L.EQ.I) GO TO 350
      IF (L.GT.I) GO TO 370
      GO TO 360
350   K1=AMOD(A(K),10.0)
      J1=(A(K)-C)/10.0
      IF (OC(J1).EQ.BLANK) GO TO 355
      OC(J1)=SYM(10)
      GO TO 360
```

```
355     OC(J1)=SYM(K1)
360     K=K+1
        IF (K.LE.MN) GO TO 340
370     I1=I
        IF(I.GT.40) I1=1
        IF (LINE.LT.50 .OR. IP.EQ.0) GO TO 375
        LINE=0
        WRITE (6,5)
375     LINE=LINE+1
        IF (IHI.EQ.MOD(I,IH)) GO TO 380
        WRITE (6,3) (CVL(J,I1),J=1,3),ST(I1),OC
        GO TO 390
380     C=BS+DS*FLOAT(I-1)
        WRITE (6,2) (CVL(J,I1),J=1,3),ST(I1),C,OC
390     CONTINUE
        WRITE (6,4) BSF,BL,CSF,BA
        RETURN
C     ERROR NOTES.
400     WRITE (6,405) M
405     FORMAT (1H0,27HNUMBER OF BASE VARIABLES IS,I3,7H.  BAD.)
        RETURN
410     WRITE (6,415) NL
415     FORMAT (18H0NUMBER OF LINE IS,I3,26H.  EXCESSIVE, NOT ALLOWED.)
        RETURN
420     WRITE (6,425) M
425     FORMAT (39H0TOO MANY BASE VALUES FOR BOTTOM ORIENT,I3,7HGT 101.)
        RETURN
        END
```